

# *JAPANIZATION*

An Introduction to Software Japanization

” Diplomarbeit ”

by

Steffen Walter Schilke

Fachhochschule Darmstadt

Fachbereich Informatik

Referenten

Prof.Dr. Norbert Krier

Prof.Dr. Hermann Deichelmann

Höchst Japan Ltd.

in Tokyo, Japan

Betreuer

Knut Zeptner

Y. Motohashi

Sommersemester 1992

Fachhochschule Darmstadt  
Fachbereich INFORMATIK  
Prof. Dr. N. Krier

09.03.1992

Vorläufige Aufgabenstellung

für die Diplomarbeit von

**Steffen Schilke**

im Sommersemester 1991/92

**Thema: Japanization - An Introduction to Software Japanization**

Fach : Softwaretechnik

Korreferent: Prof. Dr. H. Deichelmann

Institution (FHD/extern): Hoechst AG, Tokio (Japan)

Aufgabenstellung:

Im Rahmen der Diplomarbeit soll untersucht werden, in welchem Grad Softwareprodukte an lokale japanische Verhältnisse bezüglich Sprache, Schrift, sozioergonomische Aspekte und weitere Merkmale wie Datumskonventionen angepasst werden können.

Ausgabe : 23. März 1992

Spät. Abgabe: 07. Aug. 1992

Referent:



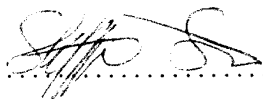
Kandidat:



Adresse: Im Grund 27  
6369 Nidderau 2  
Tel.: 06187/3963

Ich versichere, daß ich diese Diplomarbeit selbständig und nur unter Verwendung der angegebenen Hilfsmittel angefertigt habe.

Nidderau, den 7. August 1992

A handwritten signature in black ink, appearing to be 'Steffen Schilke', written over a horizontal dotted line.

Steffen Schilke

# Acknowledgements

I will and I have to thank some people who have made this work possible :

- From Hoechst Frankfurt :

Mr.Hans Hoffmann

- From Hoechst Japan Ltd. (my host company in Japan) :

Mr. G. Gerstmayer, Mr. K. Zeptner, Mr. Y. Motohashi, Mr. Yano, Miss H.Matsumoto and the very supportive staff from the I & C department.

- From the " Fachhochschule Darmstadt " :

Prof. Dr. Norbert Krier (supervisor) and Prof. Dr. Hermann Deichelmann (co-supervisor).

- Besides that I have to thank all the organizations, libraries, companies and other persons who provided me with information and their support, which has made this work possible. Also I would like to thank everybody else I tortured in the last couple of months.

The names of companies referred to herein, their corporate logos, the names of their hardware and their software may be tradenames, trademarks or registered trademarks of their respective owners and manufacturers.

© 1992 by **Steffen W. Schilke** – No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical or otherwise without prior permission by the copyright owner.

The use of " he " or " his " should be considered as sexually neutral.

## Abstract

In this thesis ( " Diplomarbeit " ) I will introduce the reader to the japanization (localization for the Japanese computer environment) of computer software systems. This means the adaptation of software to a Japanese language environment. Also I will show the main differences in the system structure. Besides that I will discuss which steps a software house should made to japanize a European or an American software product.

In the further outlook you will find an introduction to some of the japanization standardization approaches, which have been made in the last couple of years.

The project was started by world wide research in different libraries, databases and on CD ROM. Besides that I wrote several letters to companies and international organization's all over the world. After this preparation I was undertaking a research trip to Japan and stayed there as a guest of Hoechst Japan Ltd. in Tokyo. During the time of this research trip I contacted computer & software related companies and organization. Also I have made many interviews with EDP related people. After the collection of the data I wrote this evaluation of the material.

Regarding the fact that the Japanese market is the second biggest information technology market in the world. It is worth to adapt software products to this different, but homogeneous, market. Differences like different double byte character sets, Front End Processors and the Japanese culture make it difficult, but not impossible, to succeed in this market. In the last couple of years the fast developing technology made it much easier to adapt a product to this market. Also you could consider the japanization as a first step to enter the other Asian markets (adaptation to the computer environments in China, Korea, ... cause similar problems).

**Keywords :** *japanization, internationalization (I18N), globalization, localization, regionalization, kanjification, MNLS (multi national language support), Japan, DBCS (double byte character set), SBCS (single byte character set), Hiragana, Katakana, Kanji, Asia, standards*

# Contents

<b>1</b>	<b>Introduction</b>	<b>10</b>
1.1	Japanization ?!?	12
<b>2</b>	<b>The Japanese Market</b>	<b>14</b>
2.1	Japanese IT Market in the World Market	14
2.1.1	Japan vs. World	15
2.1.2	Japanese Hardware World	16
2.1.3	Japans Software World	19
<b>3</b>	<b>Doing Business in Japan</b>	<b>53</b>
<b>4</b>	<b>Cultural Differences</b>	<b>56</b>
4.1	Introduction	56
4.2	Japanese Character Sets	58
4.2.1	ASCII and Katakana	58
4.2.2	Development of Kanji Character Sets	60
4.2.3	Japanese Character Set Mess	64
4.2.4	DBCS Problems	86
4.3	Number & Currency convention	101
4.3.1	Number representation	101
4.3.2	Rounding of Numbers	102
4.3.3	Currency	102

4.4	Date & Time convention . . . . .	104
4.4.1	Date formats . . . . .	104
4.4.2	Time formats . . . . .	107
4.5	Kana & Kanji Input . . . . .	115
4.5.1	Front End Processor . . . . .	115
4.6	Minor Cultural Differences . . . . .	130
4.6.1	Writing Style Specialities . . . . .	130
4.6.2	Other Differences . . . . .	137
4.6.3	Hardware . . . . .	138
4.6.4	Sorting . . . . .	142
4.6.5	Japanese Data-types . . . . .	143
4.6.6	Japanization Pitfalls . . . . .	144
4.6.7	Cultural Differences . . . . .	146
<b>5</b>	<b>Japanization</b>	<b>163</b>
5.1	First Steps . . . . .	163
5.1.1	Adding Japanese features . . . . .	164
5.1.2	Full Japanization . . . . .	165
5.1.3	Japanese Operating Systems . . . . .	165
5.2	Japanese Software Environment . . . . .	166
5.2.1	MS-DOS . . . . .	166
5.2.2	UNIX . . . . .	168
5.2.3	MS-DOS vs. UNIX . . . . .	174
5.3	Ways of Japanization . . . . .	180
<b>6</b>	<b>Further Outlook</b>	<b>182</b>
6.1	Unicode & TAD . . . . .	182
6.2	Pen Computing . . . . .	184

<i>CONTENTS</i>	3
6.3 Asianization / Internationalization . . . . .	184
6.3.1 Asianization . . . . .	185
6.3.2 Internationalization / Globalization . . . . .	185
<b>Bibliography</b>	<b>188</b>
<b>A Index</b>	<b>191</b>
<b>B Glossary</b>	<b>195</b>
<b>C Character Sets</b>	<b>201</b>
<b>D Useful Addresses</b>	<b>208</b>



# List of Figures

2.1	Worldwide IT Market Share 1990 . . . . .	22
2.2	Worldwide IT Market Share 1991 . . . . .	23
2.3	Worldwide IT Market Share 1995 . . . . .	24
2.4	Worldwide IT Market by IDC Regions 1990 & 1995 . . . . .	25
2.5	Proportions of IT Market 1990, HW USA vs. Japan . . . . .	26
2.6	Japans Proportions of World IT Market 1990, HW . . . . .	27
2.7	Japans Proportions of Asia/Pacific IT Market 1990, HW . . . . .	28
2.8	Proportions of IT Market 1990, HW in Japan . . . . .	29
2.9	Major IT Markets 1995, Japan vs. USA . . . . .	30
2.10	Major IT Markets 1995, Japan vs. Europe . . . . .	31
2.11	Projected PC Shipment in Japan by Type . . . . .	32
2.12	PC Shipment in Japan by CPU Type . . . . .	33
2.13	Shipment Trends of PC & Laptop Computers in Japan . . . . .	34
2.14	PC Market Share in Japan . . . . .	35
2.15	PC Shipment in Japan by OS Type . . . . .	36
2.16	Pc Market share by OS . . . . .	37
2.17	Expanding OS Market Share . . . . .	38
2.18	WKS Shipment in Japan . . . . .	39
2.19	Software Products & HW Platforms in Japan . . . . .	40
2.20	Interest in foreign SW on HW Platforms . . . . .	41
2.21	Software Demand Forecast 1990's . . . . .	42

2.22	Projection of Computer Software Market . . . . .	43
2.23	Software Products Trends in Japan . . . . .	44
2.24	Type of Software Product Development . . . . .	45
2.25	Comparison of Software Products Market . . . . .	46
2.26	Software development & Programming . . . . .	47
2.27	Custom Software & Software products . . . . .	48
2.28	Software Service Market Forecast 1993 . . . . .	49
2.29	Market Share of Packaged Software, 83-88 . . . . .	50
2.30	Market Share of Packaged Software, today . . . . .	51
2.31	Top 30 Software Packages sold in Japan . . . . .	52
4.1	ASCII Table . . . . .	65
4.2	Hepburn and Nipponsiki, Page 1 . . . . .	69
4.3	Hepburn and Nipponsiki, Page 2 . . . . .	70
4.4	Hepburn and Nipponsiki, Page 3 . . . . .	71
4.5	Hepburn and Nipponsiki, Page 4 . . . . .	72
4.6	Hepburn and Nipponsiki, Page 5 . . . . .	73
4.7	Katakana Keyboard, Page 1 . . . . .	74
4.8	Katakana Keyboard, Page 2 . . . . .	75
4.9	JIS X0201, Page 1 . . . . .	76
4.10	JIS X0201, Page 2 . . . . .	77
4.11	The Joyo Kanji character table . . . . .	78
4.12	DBCS 2-byte matrices, Page 1 . . . . .	79
4.13	DBCS 2-byte matrices, Page 2 . . . . .	80
4.14	DBCS 2-byte matrices, Page 3 . . . . .	81
4.15	DBCS 2-byte matrices, Page 4 . . . . .	82
4.16	DBCS 2-byte matrices, Page 5 . . . . .	83
4.17	DBCS 2-byte matrices, Page 6 . . . . .	84

4.18	DBCS 2-byte matrices, Shift JIS . . . . .	85
4.19	Cursor Movement . . . . .	93
4.20	Backspace operation . . . . .	94
4.21	Replace function . . . . .	95
4.22	Upper and Lower casing . . . . .	96
4.23	Word wrap . . . . .	97
4.24	Line truncation . . . . .	98
4.25	DBCS windowing . . . . .	99
4.26	Search string function . . . . .	100
4.27	Japanese Kansuji numbers . . . . .	103
4.28	Emperor eras and Numbers . . . . .	109
4.29	Date representation . . . . .	111
4.30	Japanese names for the month . . . . .	112
4.31	Japanese names for the days . . . . .	113
4.32	Time representation . . . . .	114
4.33	Japanese Input Manager . . . . .	118
4.34	Photos from the " Deutsches Museum Muenchen " . . . . .	121
4.35	Hiragana Keyboard . . . . .	122
4.36	FEP conversion development . . . . .	123
4.37	Japanese and German IBM PC55 . . . . .	124
4.38	Kanji and Katakana characters . . . . .	125
4.39	JIM conversion . . . . .	126
4.40	Kana Characters . . . . .	127
4.41	Kana Characters (Part 2) . . . . .	128
4.42	Different Input Methods . . . . .	129
4.43	Japanese Writing Directions . . . . .	131
4.44	Japanese Page Order (1) . . . . .	131

4.45 Japanese Page Order (2) . . . . .	132
4.46 Japanese Page Order(3) . . . . .	132
4.47 Japanese Poem . . . . .	147
4.48 Line head processing . . . . .	148
4.49 Line end processing . . . . .	149
4.50 Rubi, Amikake and Small Kanas . . . . .	150
4.51 Dakuon . . . . .	151
4.52 Yo'on, Sokuon and Handakuon . . . . .	152
4.53 Kinto Waritsuke and Punctuation . . . . .	153
4.54 Telephone Numbers . . . . .	154
4.55 Japanese Units . . . . .	155
4.56 Japanese Units, Page 2 . . . . .	156
4.57 Japanese office Computer . . . . .	157
4.58 Japanese Keyboards, Page 2 . . . . .	158
4.59 Japanese Keyboards, Page 3 . . . . .	159
4.60 50-on Sequence and I-RO-HA . . . . .	160
4.61 50-on Sequence and I-RO-HA . . . . .	161
4.62 Radicals . . . . .	162
5.1 UNIX System V Layers . . . . .	169
5.2 NLS program skeleton . . . . .	174
5.3 Japanese Windows and Help-screen . . . . .	176
5.4 Japanese Lotus 123 and Filenames . . . . .	179
6.1 Unicode BMP . . . . .	186
6.2 TAD character set . . . . .	187
C.1 JIS Code, Page 1 . . . . .	202
C.2 JIS Code, Page 2 . . . . .	203

C.3 EUC Code, Page 1 . . . . . 204

C.4 EUC Code, Page 2 . . . . . 205

C.5 Shift JIS Code, Page 1 . . . . . 206

C.6 Shift JIS Code, Page 2 . . . . . 207

# List of Tables

4.1	Different SO/KI and SI/KO codes . . . . .	66
4.2	Support of JIS standard character set . . . . .	67
4.3	Different DBCS Implementations . . . . .	68
4.4	Number Notations . . . . .	101
4.5	Gergorian Date representation . . . . .	104
4.6	Japanese eras . . . . .	110
4.7	Time formats . . . . .	110
4.8	Demo Table for Japanese Yes / No . . . . .	137
4.9	Standard Paper Sizes used in Japan . . . . .	139
4.10	IBM vs. NEC BIOS calls . . . . .	141
5.1	EUC representations, External Code . . . . .	177
5.2	EUC representations 16-bit, Internal Code . . . . .	177
5.3	EUC representations 32-bit, Internal Code . . . . .	178
5.4	Japanese EUC mapping . . . . .	178

# Chapter 1

## Introduction

In this paper I will try to show the country specific differences, which make it difficult for European and American software companies to penetrate the Japanese software market.

The topics which are not covered in this thesis are things like import tax, legal issues, copyright problems and the distribution of software in Japan. The main emphasis of this thesis is the software technical aspect in the japanization of foreign software. In order to do this I divided the thesis into the following chapters :

### 1. Japanese Market

A presentation of current state and a future outlook of the Japanese hardware & software market. I also will give some information about the percentual spread of hardware & software and the type of computer power usage in Japan. Besides that I will discuss possible trends in the usage of computer systems in Japan.

### 2. Doing Business in Japan

A short overview about the Japanese way of doing business. Pitfalls and tips to get a long in the Japanese Business world.

### 3. Cultural Differences

One of the major parts of my thesis is the introduction to the Japanese cultural specialities e.g.,

- writing (Hiragana, Katakana, Kanji) and the arising problems e.g., storing this data, keyboard input, keyboard layout, the character set, ...
- date, time and currency conventions
- number representation and units
- Japanese standards in computing
- cultural depending differences
- ...

This chapter is divided into three parts :

- (a) major cultural differences (language dependent, character set, ...)
- (b) minor cultural differences (date & time format, numbers, ...)
- (c) localization pitfalls

#### 4. Japanization

Here I will introduce the reader to possible solutions for the japanization of computer software products. I will discuss existing approaches like the OADG DOS/V or Microsoft Windows, the NEC computer line. Also I will have a look on future UNIX system standards.

#### 5. Further outlook

Here I will discuss the impact of japanization (localization) and will try to present a further outlook of the impacts of the localization of computer software systems (e.g., Korea, China, ...)

#### 6. Bibliography

It includes the books which I used as reference material.

#### 7. Appendix

Here you will find an index, the glossary and some examples of the different character sets.



## 1.1 Japanization ?!?

Let me now start with an introduction to the arising problems if you want to adapt software for the Japanese market.

First of all it is possible to think " Why should I adapt my software to the Japanese Computer Environment ! ". If so you should stop reading this paper. Now ask yourself how *you* would react if some one would try to sell a software product to you with a catalog, manual, help screens or program messages written in Tagalog, Hindi or Japanese. If you say " I would not buy this product " then you got exactly the reaction of a Japanese computer user. If you would try to sell him an, e.g., English or German software product with native language messages or manuals.

It is true that for some limited purposes it is possible to sell English versions of a program all over the world, but this covers only the fields like operating systems, communication systems or other products used by EDP specialists. The normal everyday user expects software with help screens, messages and manuals in his own native language. Adapted to his, e.g., Japanese computer environment.

For the Japanese it is not very convenient to read their language written in our roman alphabet. Furthermore it could cause misunderstanding. It is possible to write Japanese names or addresses in roman characters, but to write a full Japanese text in roman characters would make this text very difficult to read. Output in the roman alphabet, e.g., printouts or invoices, would make the customers upset. If you want to send product information's to your customers it is essential to provide this information in the native language of the customer. One of the last, but quite important, points are that if you have to file documents to governmental authority you have to use the official country language and habit.

If the user is not familiar with, e.g., English, his will to explore all possibilities of the program are not very high and so the next time he will recommend to buy a product in his native language or with support in his own language.

A software vendor has to be aware that, in a foreign country, he has not only to compete with the local software vendors. He also has to win the trust of the foreign user and has to show his commitment to the foreign language version of his product. Not only quality and functionality are important, also how good the adaptation to the, e.g., Japanese Computer Environment is, is very important. To get a share of a

foreign market the foreign software vendor has to win over the local products. The only way to do this is to be better as them. This implies that the product *must* support country specific features. To do this a software vendor needs a certain knowledge of the computer, cultural and language environment in the target country.

With this paper it should be possible to get a first overview about the " Things to do " to create a Japanese version of a software product.

# Chapter 2

## The Japanese Market

In this chapter I will present an overview of the Japanese information technology (in future called : IT) market. This market is some type of unique marketplace in the world. The structure is different from the market in USA or Europe. The position of the Japanese IT market in the world IT market and the diverse Japanese hardware market will be presented first in this chapter. In this part I will set the main emphasis on the PC/Workstation (in future called : PC/WKS) market. After that I will present an overview of the Japanese software market (Also mainly for the PC/WKS environment). All statistical data are gathered from different sources like IDC, JCQ, JPL, Softic, Miti, JISA, AEA, JETRO and many more. The problems of gathering these data are, e.g., that the Japanese fiscal year runs from April to March, the main information is only available in Japanese and that most English data are at least 1–3 years old.

### 2.1 Japanese IT Market in the World Market

The Japanese IT market is the second biggest IT market in the world. This is supported by the fact that the Japanese IT market is a homogeneous market. Unlike in the English speaking countries, where you do not have a homogeneous market (e.g., American, British and Australian English, different terms for technical denominations, different cultural backgrounds, ...). The Top 5 players in the world IT markets are USA, Japan, Germany, France and the UK. The rest of the world market (in future called : ROW) has a joint market share below 30 %, as it is a very diverse market

with more than 30 countries and even more different languages as well as cultures. Each country of this group holds a market share below 4 %, more often below 1 %.

### 2.1.1 Japan vs. World

If we look at the World IT Market 1990 (see page 22, figure 2.1,[32]) we will see the following market situation (regarding IDCs Worldwide IT Market survey) :

- USA 36.2 %
- Japan 18.1 %
- Germany 7.9 %
- France 6.7 %
- UK 5.9 %

The US market is about twice as big as the Japanese market and the Japanese market is more than twice the size than the follow-up market Germany. This fact alone is quite interesting, but if we contemplate the development of the world IT market we will realize that this fact is getting more important in the future. IDC's data for 1991 (see page 23, figure 2.2,[32]) show that the Japanese IT market is the only market with a visible growth rate. The other Top 5 markets, with exception of Germany, have a loss in their world IT market share. This gets even more dramatic if we look at the predicted market shares for 1995 (see page 24, figure 2.3, [32]). The US market share will shrink to 33.9 % which will be just 1.5 times the size of the Japanese market share. Furthermore (compared to the follow-up market Germany) the Japanese share has increased to 2.75 times the German market share. If we take a look at the market shares by IDC world regions (see page 25, figure 2.4,[32]) we will see that the European market share will decrease from about twice as big to about 1.5 times the size of the Japanese market share, but the European market is not a homogeneous market. This fact gives the Japanese market a big plus and that's why the US and Japanese market are the most important markets in the IT world, holding more the 50 % of the world IT market shares.

### 2.1.2 Japanese Hardware World

Besides the culture and language dependent differences, also the structure of the market in Japan is also different from the rest of the world's IT market. The Japanese IT market is (or was) more oriented to mid-size and large computer systems (see page 26, figure 2.5, [32]). Compared to the market in the USA this fact is quite significant. The 1990 US market share of PCs and Workstations was more than 50 %, compared to 35 % on the Japanese side. If we have a look at the Japanese PC world market share (see page 27, figure 2.6, [32]) we will see that this share was just about 13.8 % in 1989. This market share will significantly increase. When we compare this to the growth rate of small, medium and large systems it will be visible that the PC/WKS market will become quite valuable to about 20 % of the PC world market share. To see how important the Japanese market place is we have to consider the Japanese market share in the Asia/Pacific region (see page 28, figure 2.7,[32]). This makes evident that the Japanese market, with a market share among 70 to 90 % per system class, is the key market in this area. If a software product is successful in Japan this position could be used to penetrate the other markets in the Asia/Pacific region. (Besides that it is easier to adapt a japanized product to other Asian language environments, because the main work is already done, but I will explain this topic later). The future development of the Japanese HW market will change dramatically. Instead of small, medium and large computer systems the PC/WKS will win the biggest HW market share in Japan (see page 29, figure 2.8, [32]) with about 20 % of the IT market share. This is about twice as big as the large systems market share. The PC/WKS is the only systems group with an increasing market share in the next couple of years. Again let us look at the estimated US and European spread of system types compared with Japan (see page 30, figure 2.10 and page 31, figure 2.10,[19]) in the near future 1995. We will recognize the Japanese PC/WKS market share is still about 13 to 15 % smaller than in USA or Europe. Nevertheless it is the fastest growing market in the world.

#### PC/WKS Market

I will now give a brief overview about the Japanese PC/WKS market. This rapidly growing market has a fast changing structure. In the year 1990 the biggest PC/WKS

market share was represented by the combined WKS market share with more than 64 % followed by Notebook and Laptop computer types (see page 32, figure 2.11, [19]). This will change until 1995 to a 55.5 % market share for Notebooks compared to a combined 38 % market share for WKS. This does not imply that the Notebook or Laptop computers are low power machines. It shows that the future demand for HW and the way of HW development in Japan will go towards small, highly integrated, sophisticated, powerful machines. These fit the Japanese working environment where the demand to save office space is very high, because office rents in Japan are sky-high and consequently the office space is limited. This is supported by the fact that the Japanese users prefer high power, fast and reliable machines (see page 33, figure 2.12,[19]) equipped with fast processors. Since manufacturers are able to wrap this high speed technology in smaller cases, which provide the same power as a desktop type PC, the sales of Laptops (and Notebooks) are rapidly increasing (see page 34, figure 2.13,[33]).

### **IBM vs. NEC**

Unlike outside Japan, where IBM / IBM-Clones have the biggest market share in the PC world (or at least IBM has set the rules belonging the system structure), the Japanese PC market is dominated by the Japanese computer manufacturer NEC (Nippon Electronic Corp.). This fact causes, besides the culture and language dependent differences, the biggest problems for foreign software vendors. The NEC and the compatible machines from Epson hold a market share close to 60 %. IBM however, which has, since years, dictated the PC/WKS market in the USA and Europe, has a market share as small as 7 %. The other big PC/WKS manufacturers which are represented in the Japanese market are Apple, Toshiba and Fujitsu. Each of these manufacturers has their own, incompatible system and OS to the other manufactures (see page 35, figure 2.14,[23]).

### **OS in Japan**

Regarding this fact it is interesting to see that in the year 1990 MS-DOS had the biggest market share in shipped PCs (see page 36, figure 2.15,[19], Later I will talk about the problems concerning the incompatible machines and OS). If we look at the

estimated figures for the year 1995 we will see that MS/DOS will still hold a market share of more than 75 % (down from 83.6 %) that, compared with their market share 1990, UNIX and OS/2 will significantly increase their market share (see page 37, figure 2.16,[10]). In a survey from JPL 64.8 % of the questioned people believed in an increase in the market share of UNIX, with close follow-ups by the MS/DOS based MS-Windows and other MS/DOS type operating systems (see page 38, figure 2.17,[34]). This trend is supported by the fast growing numbers in WKS shipment in Japan, which are mainly driven by the UNIX or UNIX like Operating Systems (see page 39, figure 2.18,[19]) In a survey (see page 40, figure 2.19,[38]), done by SOFTIC, it is shown that the Japanese market for software products is split into three parts

- Systems Software (35 %)
- Business Applications (42 %)
- Dedicated Applications (23 %)

and that Business Applications hold the top position with 42 % of the market. Combined with Dedicated Applications the applications market share is 77 %. Furthermore you can divide the HW Platforms, which are running this software, into three parts

- Mainframe (22 %)
- General purpose Mini (small & medium type) (16 %)
- PC/WKS (62 %)

Again it is easy to recognize that the biggest share belongs to the PC/WKS type of systems. In this big market the JPL institute did a survey about the question if Japanese computer users are interested in foreign software and if so, on which HW platforms they would like to run this software (see page 41, figure 2.20,[34]). More than 70 % of the questioned users are interested in foreign software. Again UNIX gained a big share nearly equal to MAC and IBM PC. The IBM is in the lead with a 6.6 % higher share.

According to the indicators it is visible that there is an increasing PC/WKS market in Japan so that this raising market should be considered as one of the main software markets in the future. In the next section I will give an overview about the Japanese software market.

### 2.1.3 Japans Software World

In the 1989 Miti survey " Software Demand Forecast " the Japanese software market is predicted to have a high, increasing potential. The " mighty Miti ", which is the control organization for the Japanese industries, estimated the growth of the Japanese software market in a survey from 1989. From 1990 to the year 2000 the demand for software will raise to a market potential which is 4 times bigger than the actual market from 1990 (see page 42, figure 2.21,[34]). The same figures are estimated for the computer software market. There is a tremendous high growth rate expected over the next couple of years (see page 43, figure 2.22,[35]). Compared to the US and Europe the demand for Software Products is considered as a fast growing part of the Japanese IT market (see page 44, figure 2.23, [36]). Here the expectations are even higher. JISA is expecting over 4 times the market potential of 1989 in the year 2000.

#### Software Japanese Style

These forecasts sound quite good but to understand the Japanese software market you have to know how software is made in Japan. Today the biggest part of software development in Japan is still done independently. This is caused by the need for highly customized software systems in Japan. Compared to the market share of domestic developers and imported software this market share is around 1.5 times bigger and will rise towards twice the size of the combined market share of the two other fields. The two other sectors are also gaining a good share of the market leaving plenty of room for japanized software (see page 45, figure 2.24,[36]). Another surprising fact about the Japanese software product market is that the structure is very similar to the structure of the American software product market. There are only slight differences in the percentage of the different fields, also here the development trend of these markets is similar. In the year 1995 the highest market share in Japan belongs to application solutions (42 %), followed by Application tools (25.2 %). Similar to the USA the system/utilities market share will significantly decrease to 32.8 %. Let us have a look at the spread of custom software and software products in Japan (see page 47, figure 2.26,[36]). In the year 1989 the customized software market was about 6 times bigger than the market for software products. Again the reason is or was the high demand for customized solutions in the Japanese IT environment. Even if



the market share of software products will raise, it will not reach the size of the customized software market in the year 2000. Even then the market share of custom software will be 3.75 times bigger than the market share of software products. If we contemplate the sales figures which are published by JISA (see page 48, figure 2.27,[36]) we will see that the sales figures are much bigger in the custom software field than in the software product's field. Furthermore the growth rate of custom software is higher than the growth rate of software products. In the market forecast for software services 1993 by IDC we will notice that most fields will nearly double their sales and the system integration sales will be 4 times bigger than 1988 (see page 49, figure 2.28,[19]).

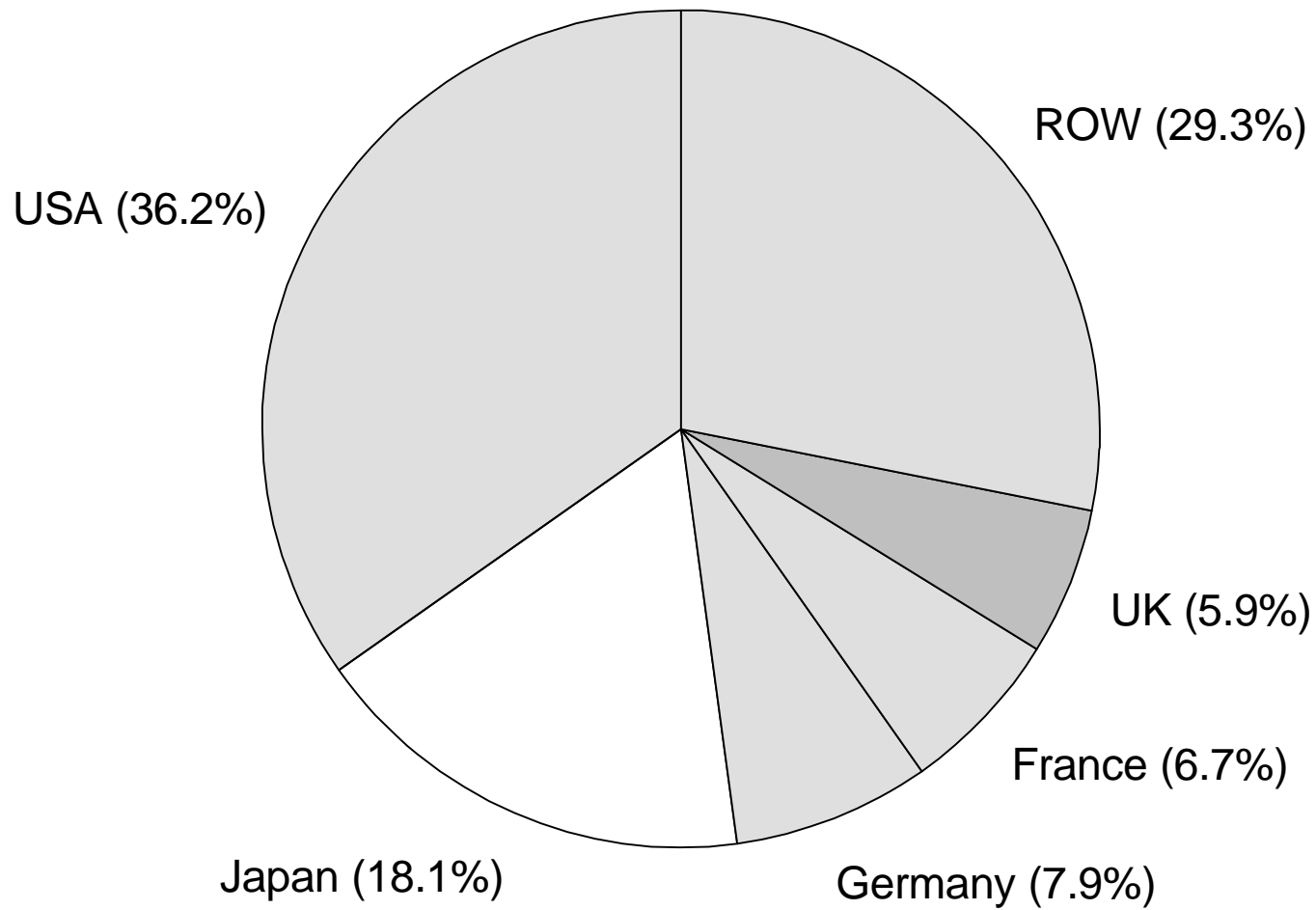
### **Package SW vs. Custom SW**

It is easy to see that the market for custom software services is more than twice as big as the market for packaged software. When we compare the historic growth of this market share between the Top 5 IT nations we will see that the Japanese share is far below the share of packaged software in other countries (see page 50, figure 2.29,[37]). The market share of packaged software in Germany 1988 was close to 50 % compared with a Japanese market share of 8 %. It is understandable because the custom software market share in Japan is so huge. Today the market share for packaged software in Japan is somewhere between 10 and 20 %, depending on the source of information. This is still small compared with the market share of packaged software in the USA (60–75 %) and Europe (40–60 %) (see page 51, figure 2.30, [10]). The last, but not least, point is the analysis of the Japanese PC/WKS packaged software market. In an IDC survey from 1990 the market share of the Top 30 selling software packages in Japan was dominated by office related programs like databases, Japanese word processors, spreadsheets and accounting software (see page 52, figure 2.31,[19]). These four software types represented about 69 % of the software packages sold in Japan. An interesting fact is that game software would be under the Top 3 software products if you considered it as a software package.

### **A Growing Market**

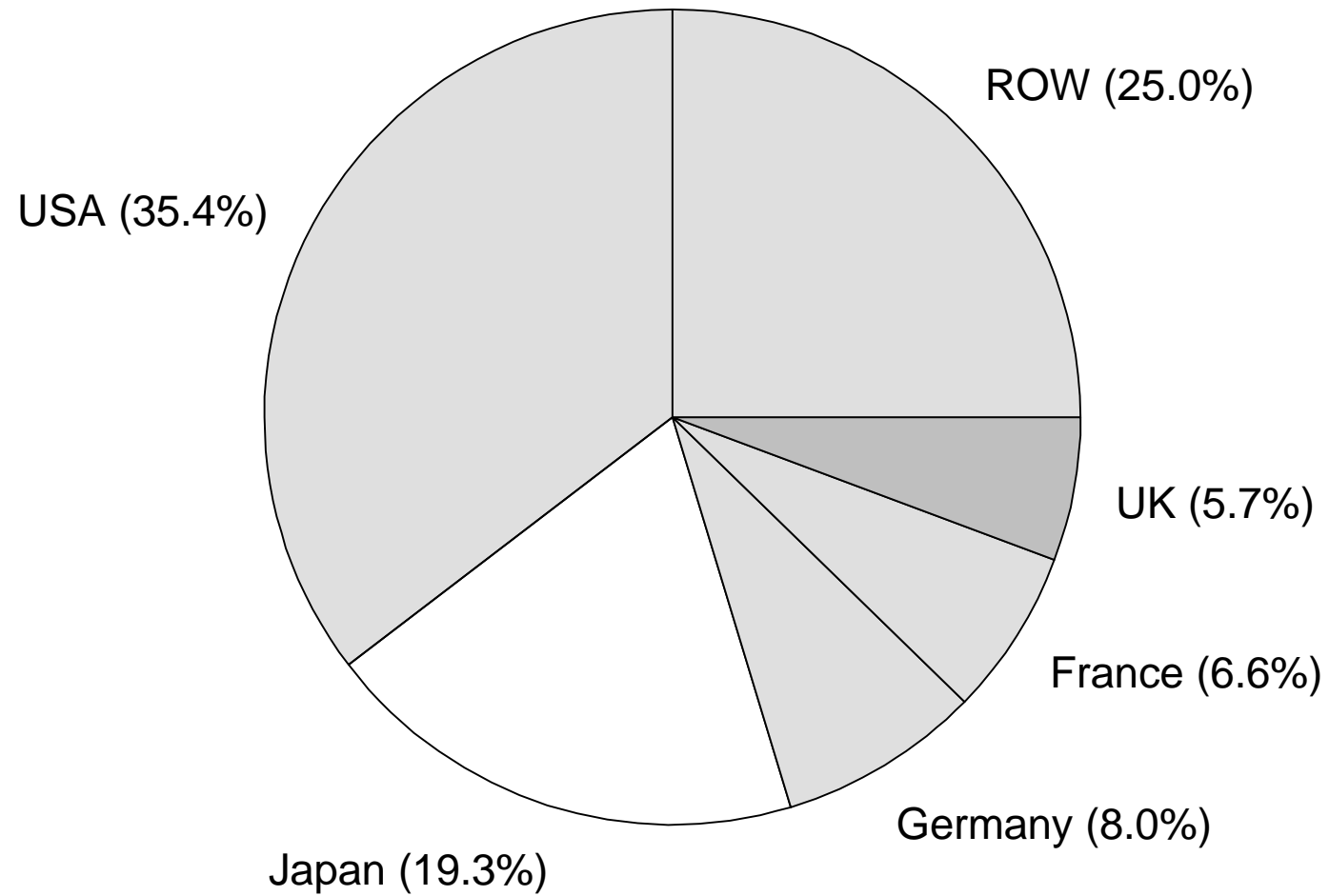
The Japanese software industry looks as if it was the only growing homogeneous market in the next couple of years. Especially the booming PC/WKS with the rocketing notebook market seems to be an attractive market for software services, products and packages. Even if it is difficult to launch a product at the beginning, the reward will be even higher. Besides the decreasing American market the Japanese market has to be considered as one of the big chances for software companies today.

## Worldwide IT Market Share 1990



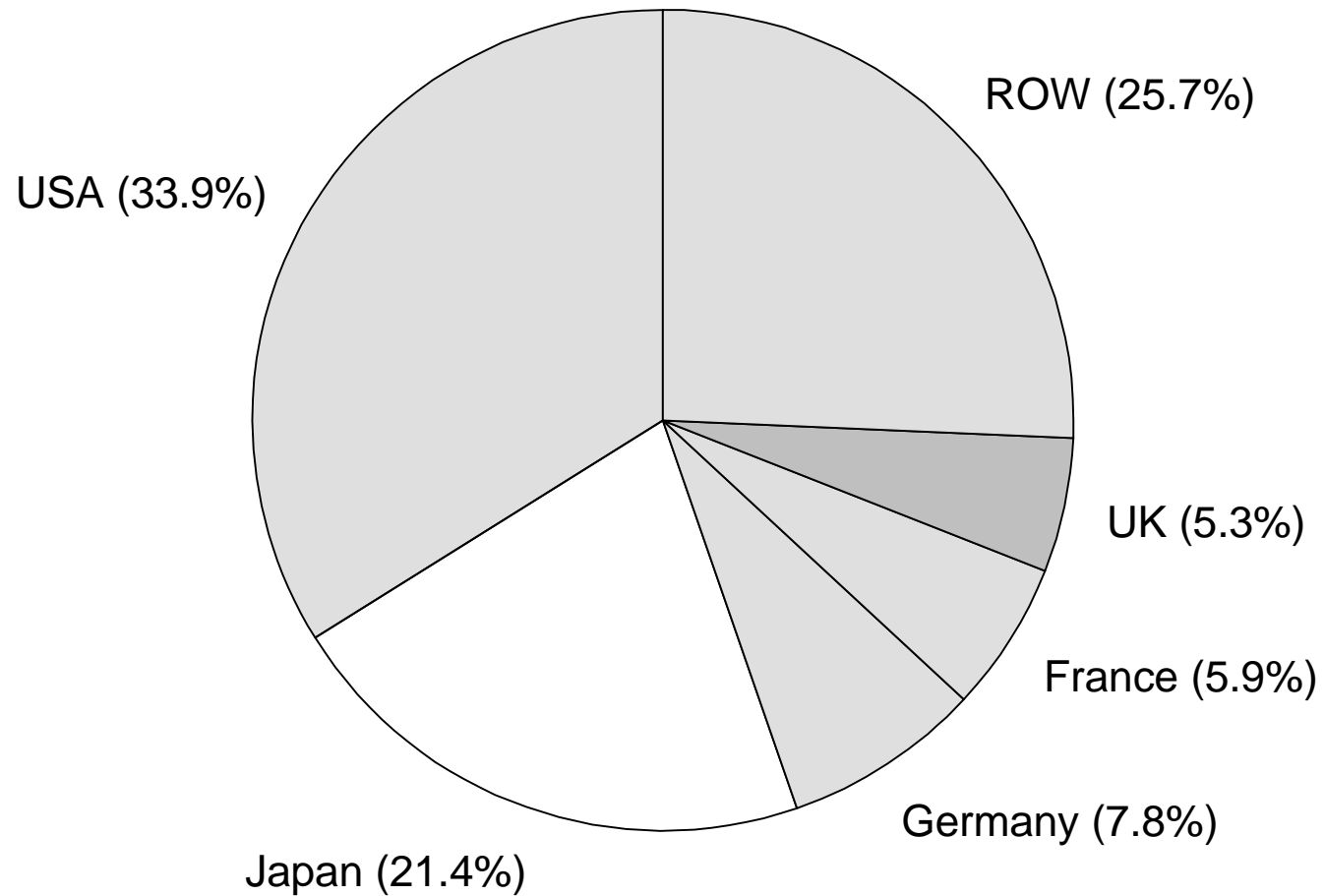
Source: IDC, 1991

# Worldwide IT Market Share 1991



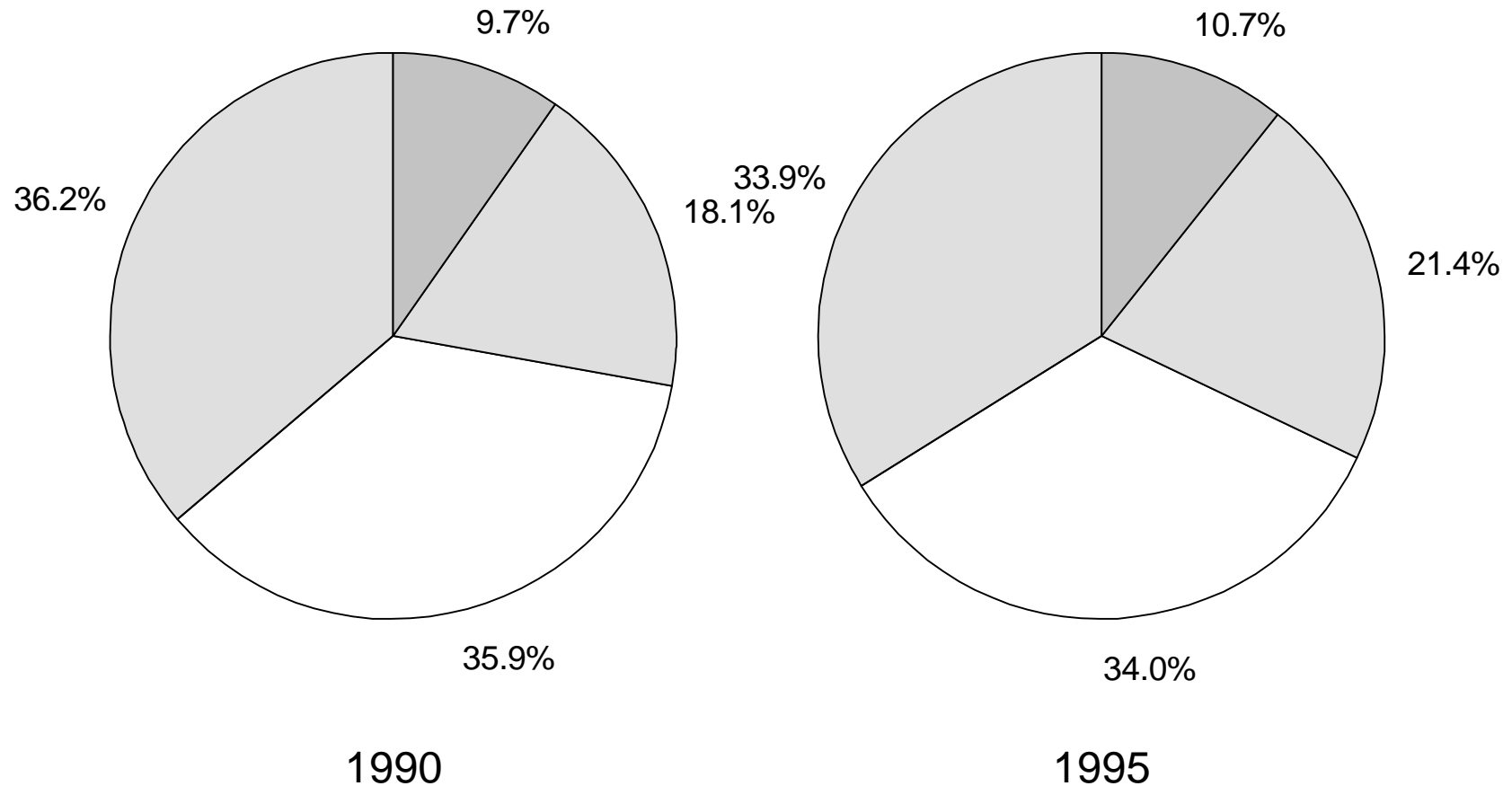
Source: IDC, 1991

## Worldwide IT Market Share 1995



Source: IDC, 1991; estimated

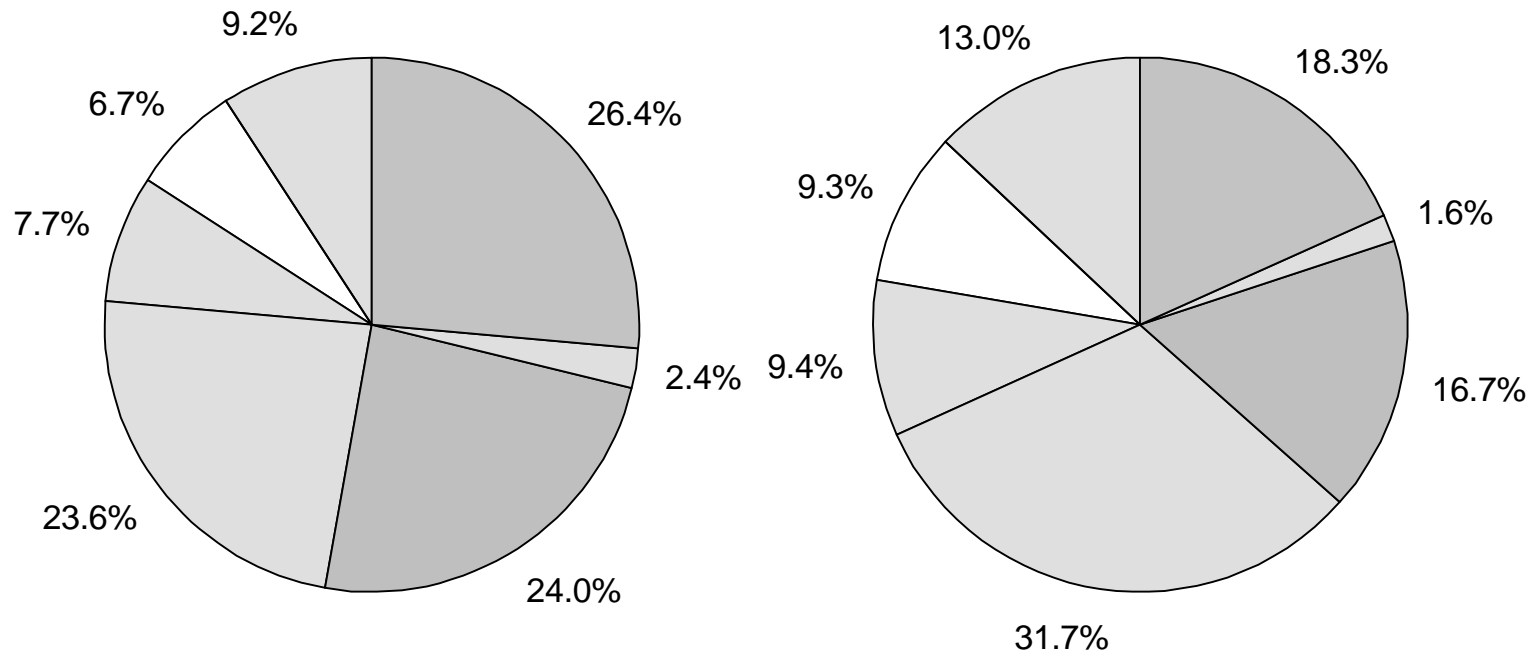
# Worldwide IT Market by IDC Regions



USA Europe Japan ROW

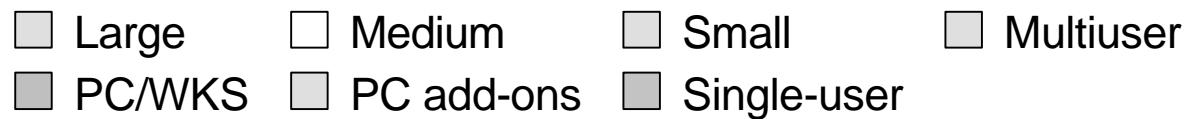
Source: IDC, 1991, percentage of World IT Market

## Proportions of IT Market 1990 Hardware in USA vs Japan



USA

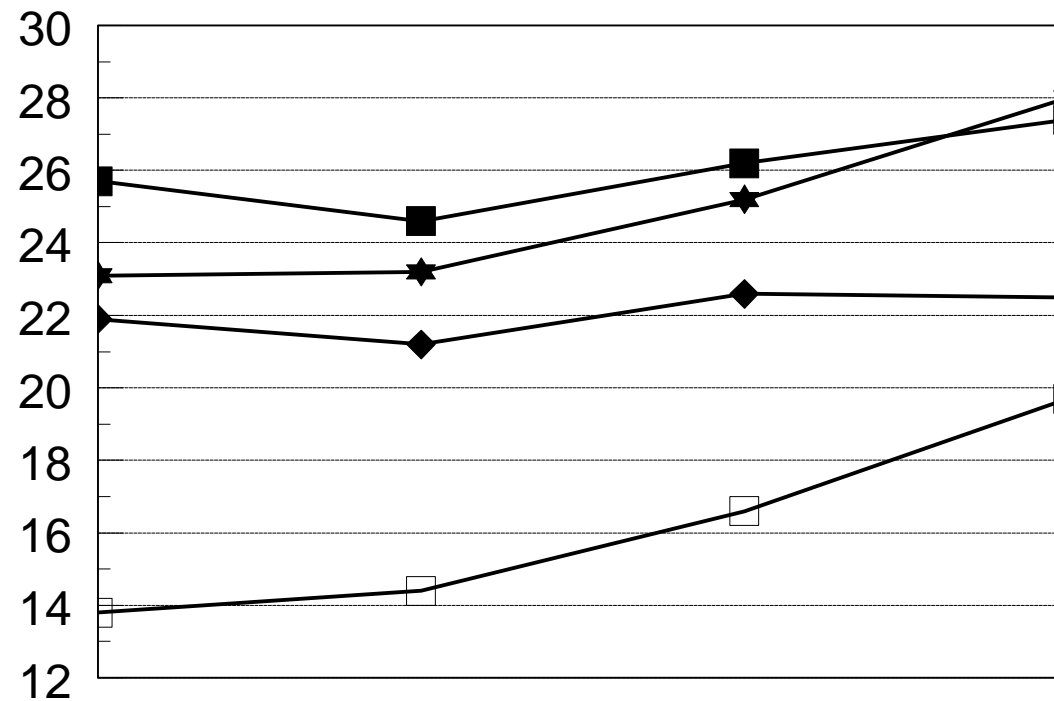
Japan







Source: IDC, 1991

# Japan's Proportions of World IT Market 1990

## Hardware

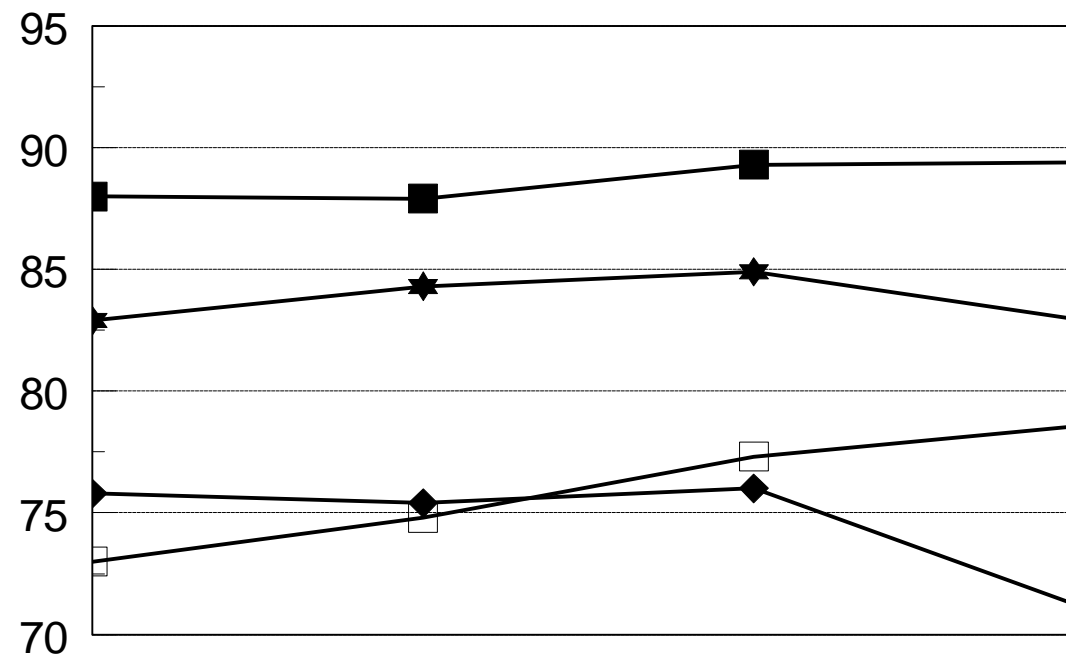


	1989	1990	1991	1995
Large 	25.7	24.6	26.2	27.4
Medium 	21.9	21.2	22.6	22.5
Small 	23.1	23.2	25.2	28.0
PC/WKS 	13.8	14.4	16.6	19.7

Source: IDC, 1991, percentage; forecast



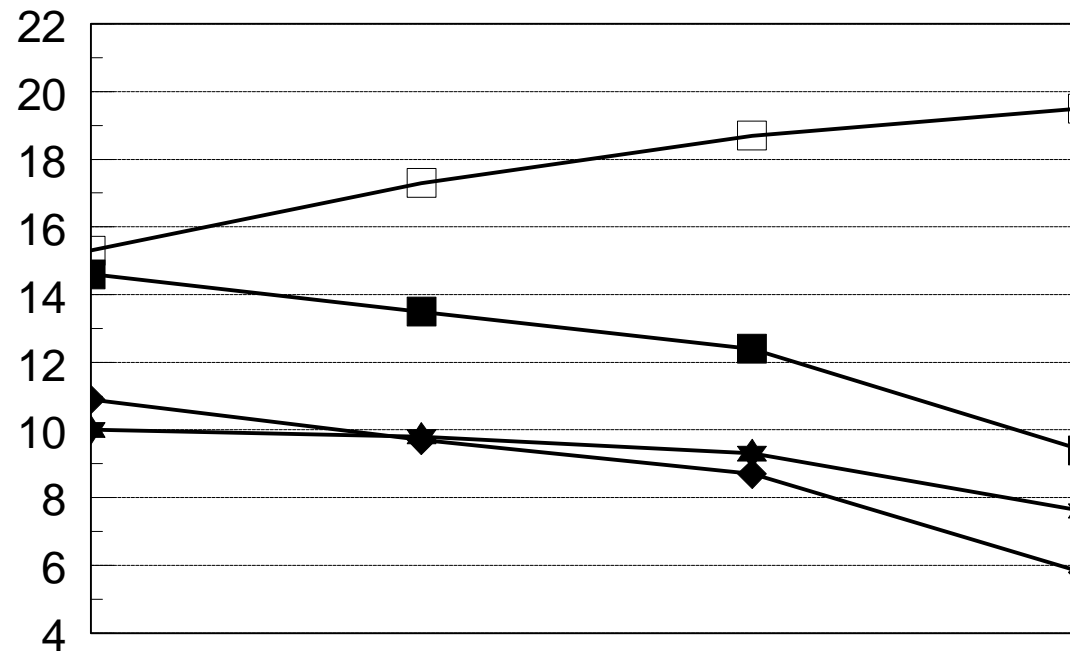
## Japan's Proportions of Asia/Pacific IT Market 1990 Hardware



	1989	1990	1991	1995
Large <span>■</span>	88.0	87.9	89.3	89.4
Medium <span>◆</span>	75.8	75.4	76.0	71.1
Small <span>★</span>	82.9	84.3	84.9	82.9
PC/WKS <span>□</span>	73.0	74.8	77.3	78.6

Source: IDC, 1991, percentage; forecast

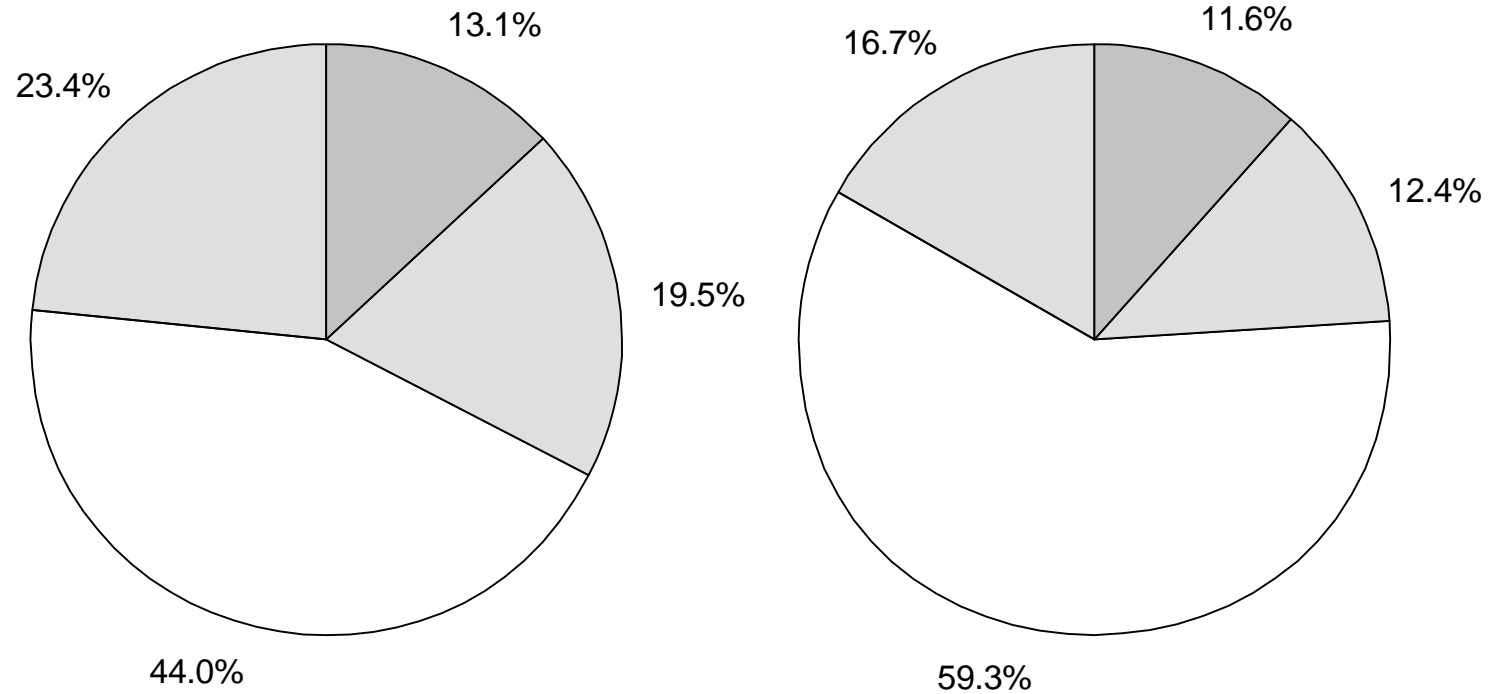
## Proportions of IT Market 1990 Hardware in Japan



	1989	1990	1991	1995
Large <span>■</span>	14.6	13.5	12.4	9.4
Medium <span>◆</span>	10.9	9.7	8.7	5.8
Small <span>★</span>	10.0	9.8	9.3	7.6
PC/WKS <span>□</span>	15.3	17.3	18.7	19.5

Source: IDC, 1991, percentage; forecast

## Major IT Markets 1995 Japan vs. USA



Japan

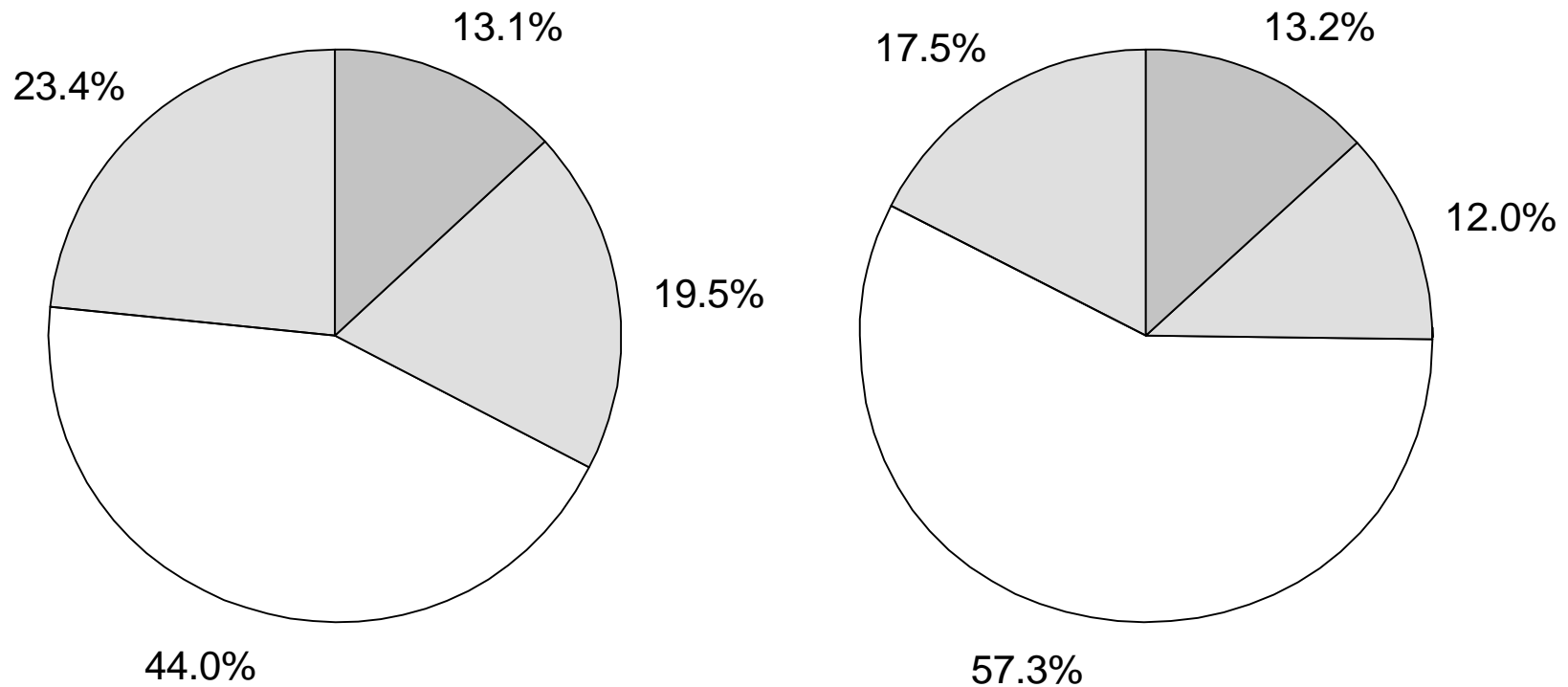
USA

Large PC/WKS Small Medium

Source: IDC, 1991, percentage; forecast

# Major IT Markets 1995

## Japan vs. Europe



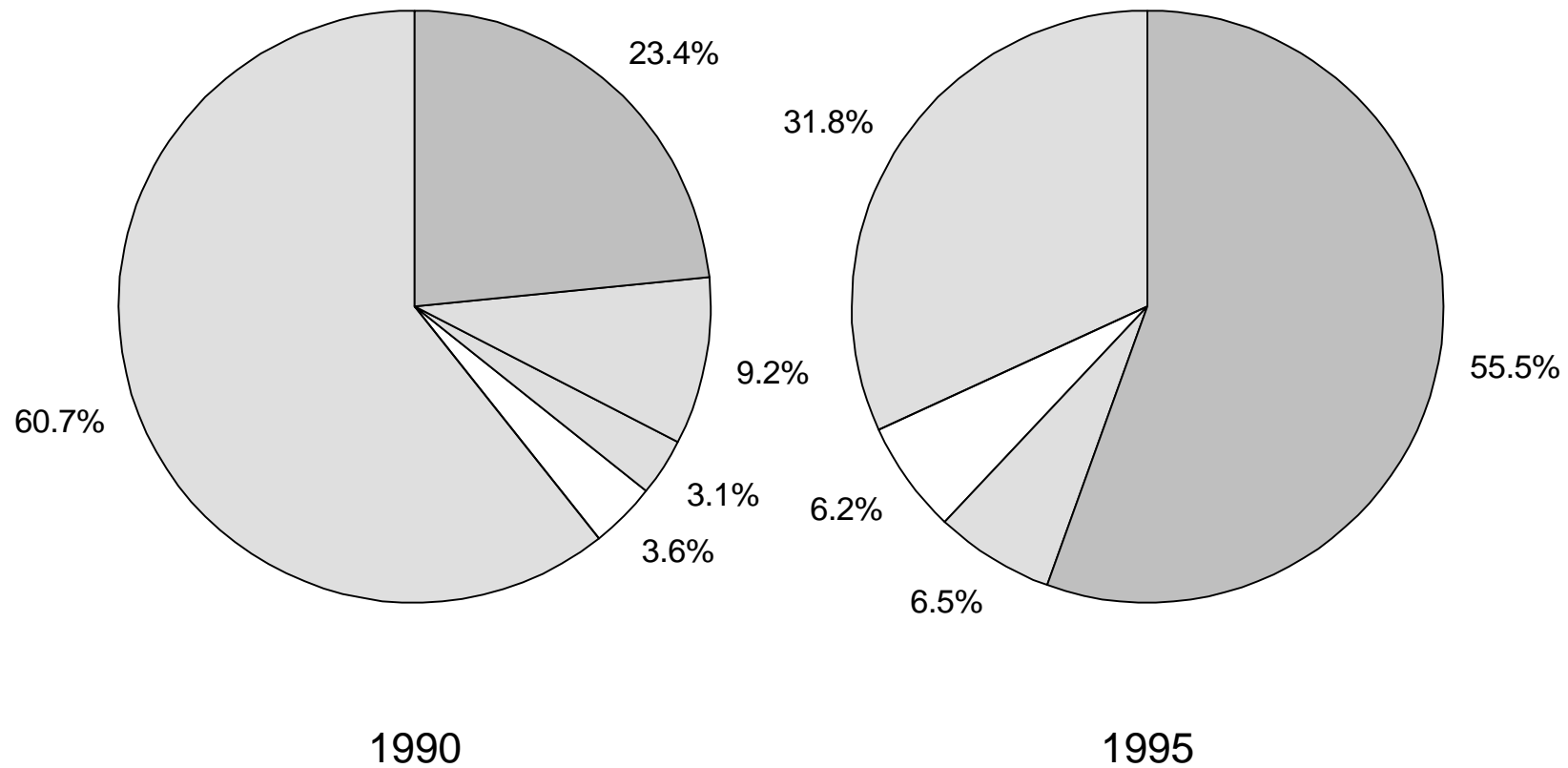
Japan

Europe

Large PC/WKS Small Medium

Source: IDC, 1991, percentage of IT Market

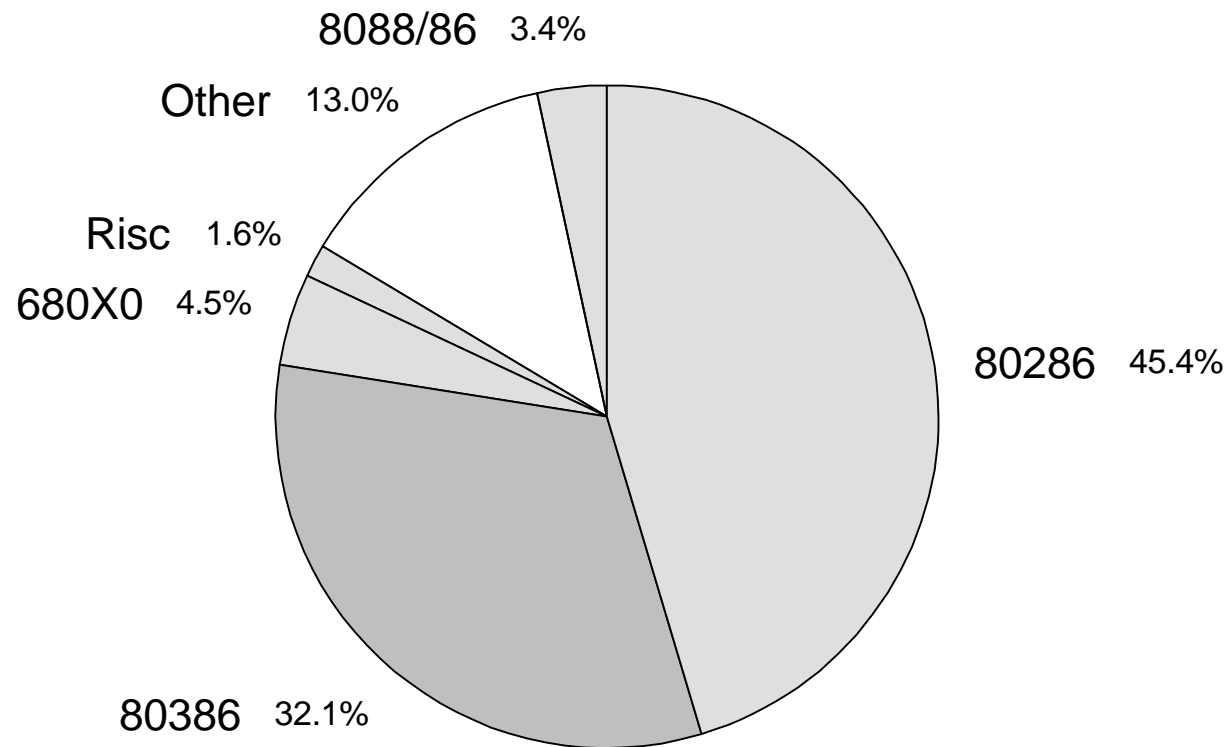
## Projected PC Shipment in Japan by Type



Office WKS Engineering WKS Others Laptop Notebook

Source: IDC, 1991; forecast for 1995

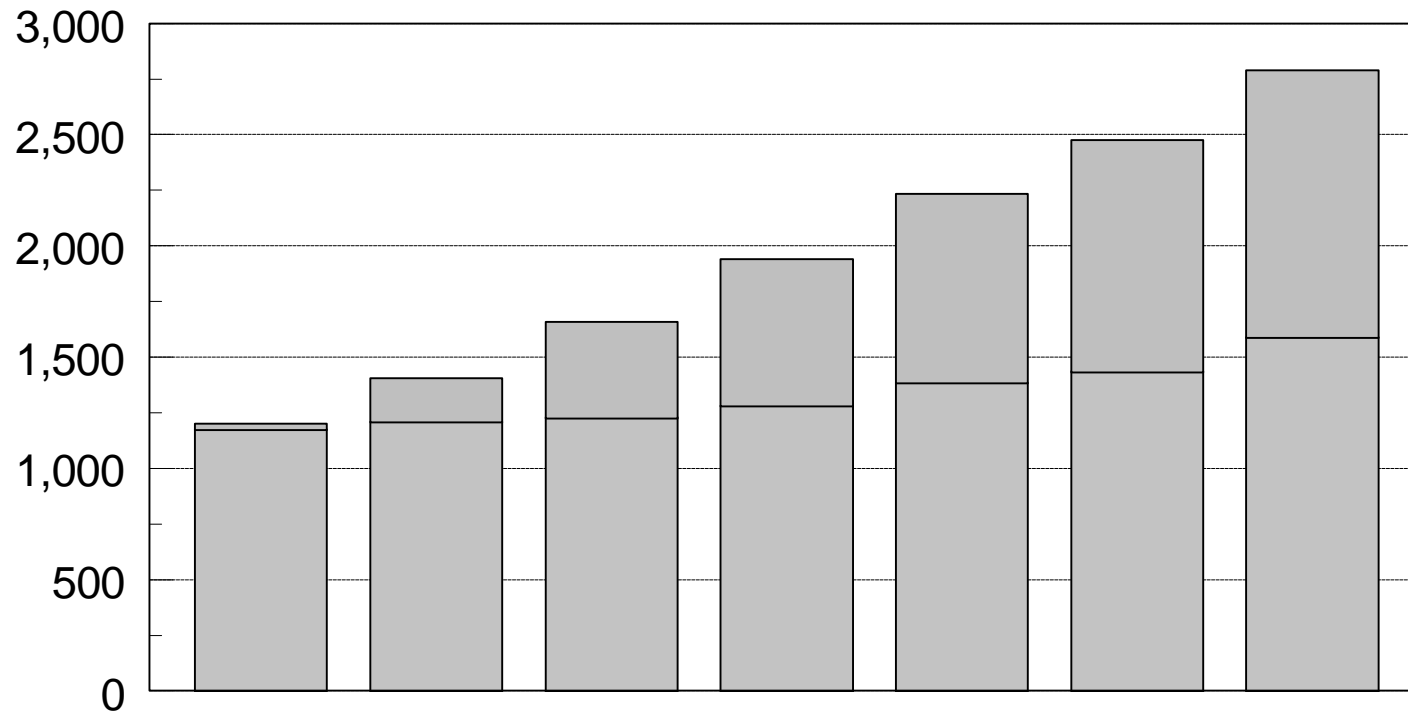
## PC Shipment in Japan by CPU Type 1990





CPU type

Source: IDC, 1991, in percentage

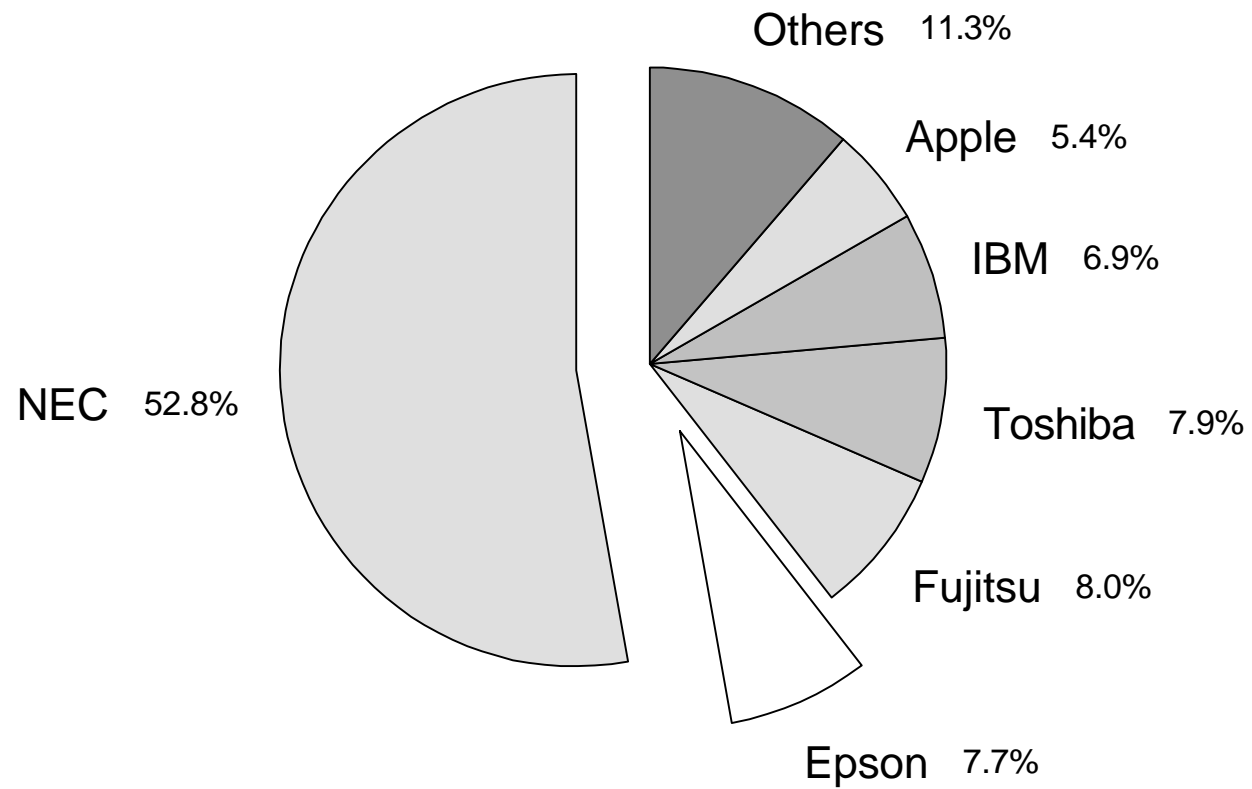
## Shipment Trends of PC & Laptop Computers



	1987	1988	1989	1990	1991	1992	1993
PC 	1,176	1,210	1,229	1,281	1,386	1,436	1,590
Laptop 	24	197	432	660	850	1,040	1,200

Source: JCQ, per 1000 Computers; forecast

## PC Market Share in Japan

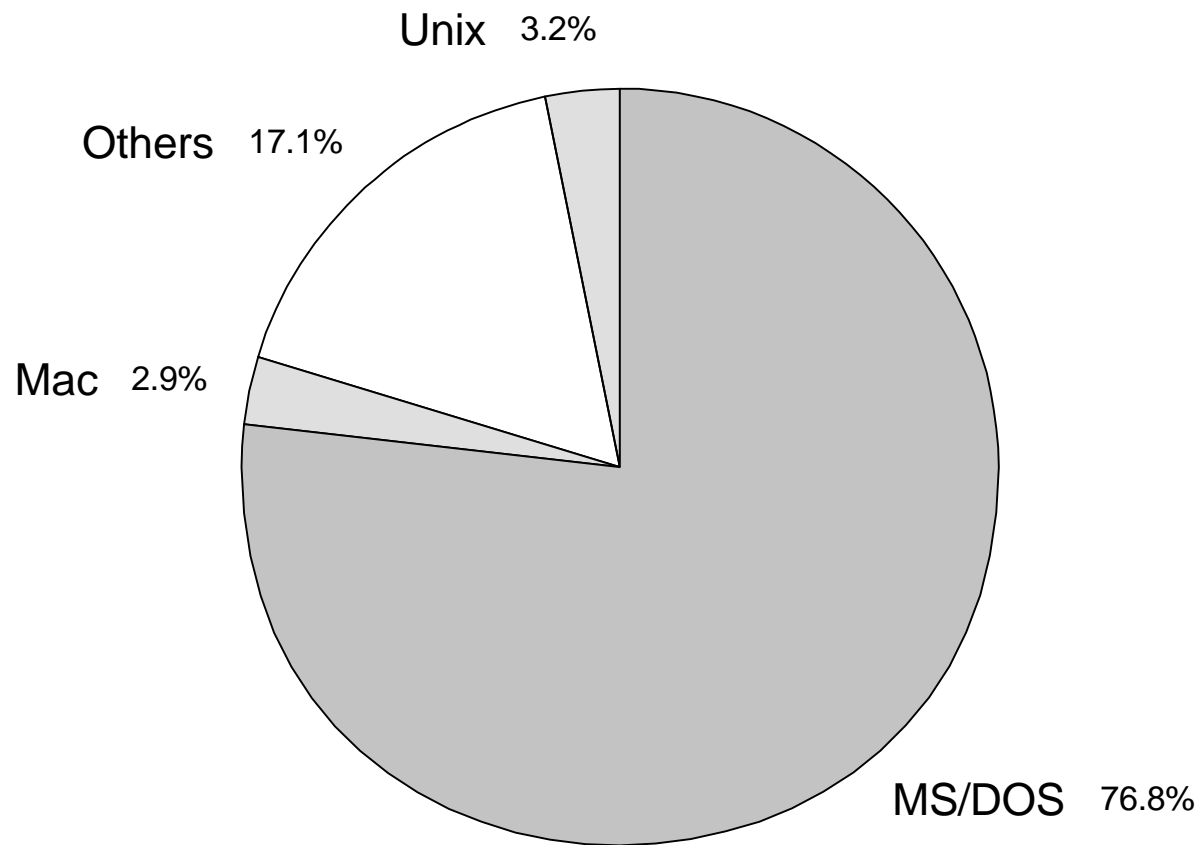


1991

Sources: NEC, in percentage; estimated

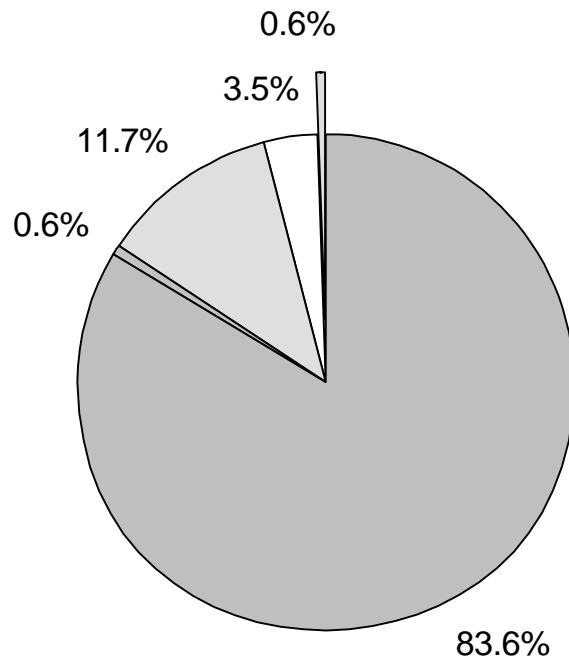


## PC Shipment in Japan by OS Type 1990

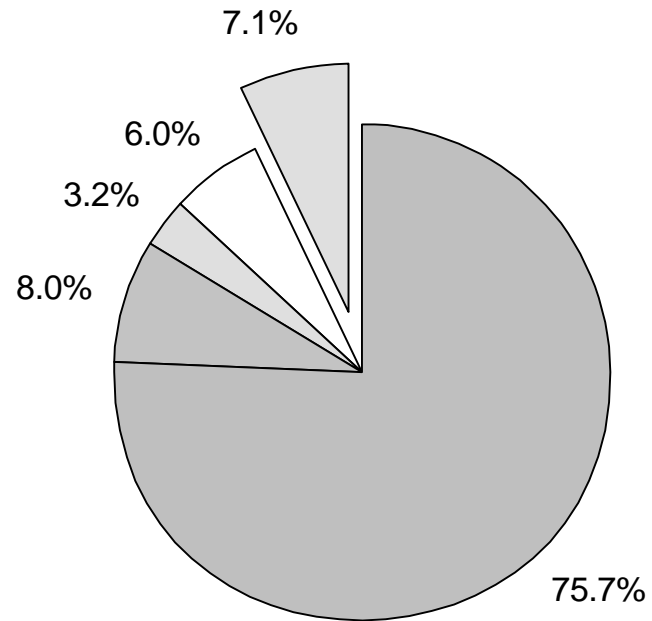


Source: IDC, 1991, in percentage

## PC Market by OS



1990

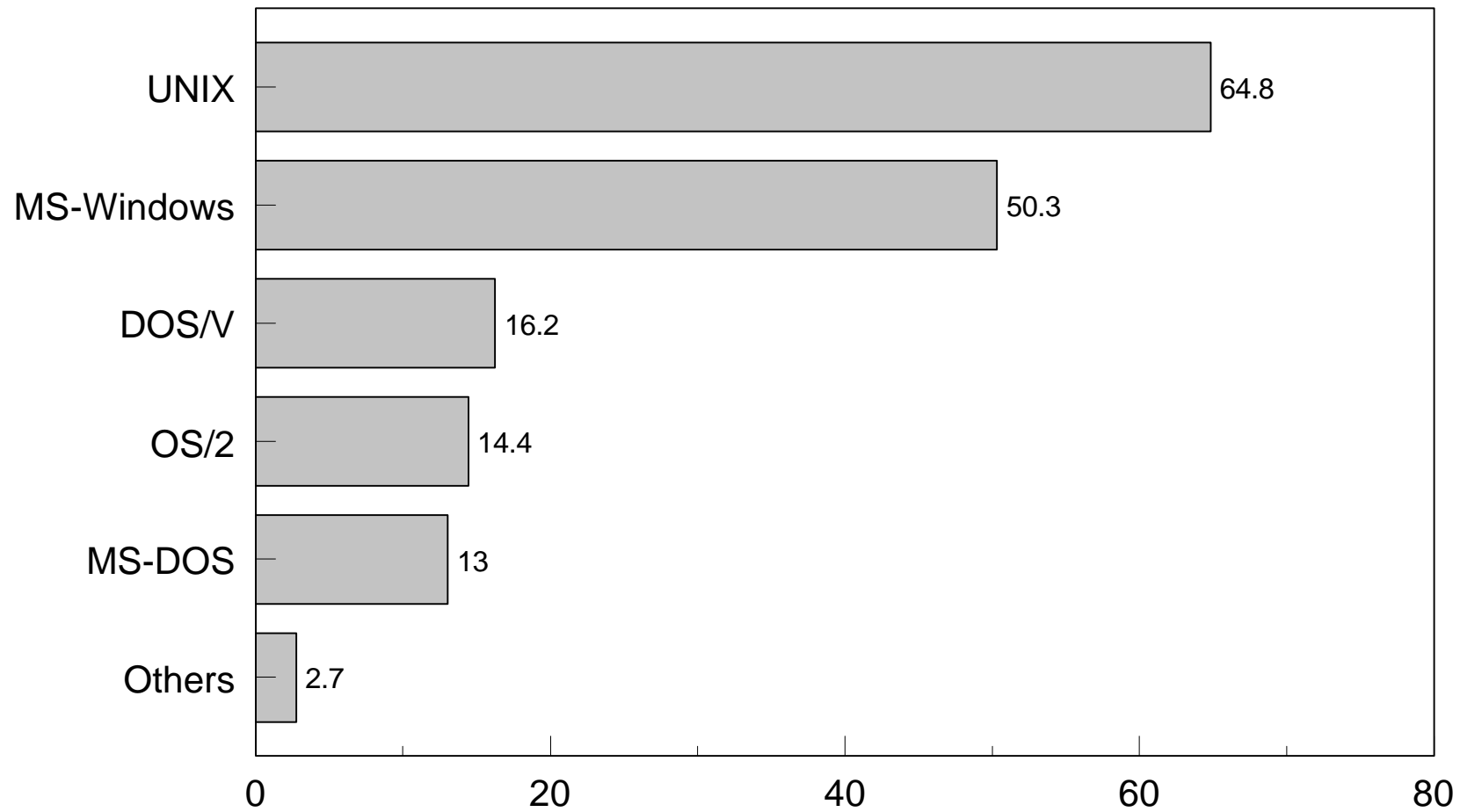


1994

OS/2 MAC Others UNIX MS/DOS

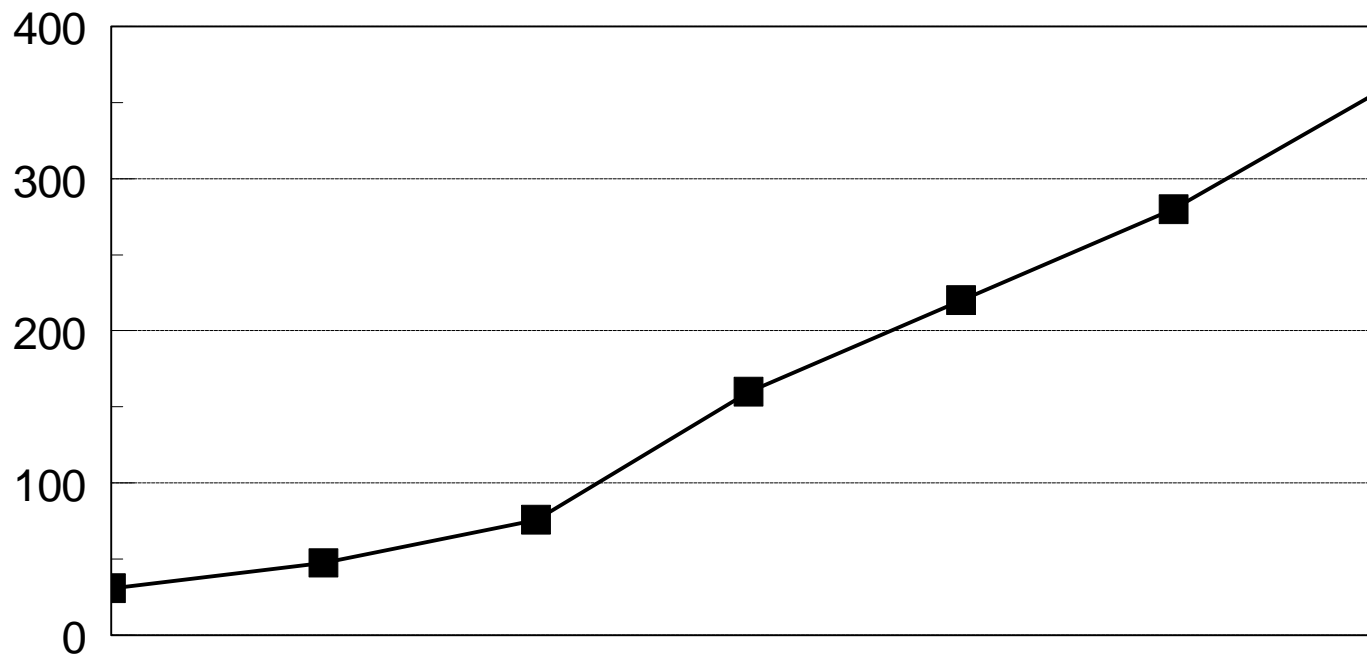
Sources: AEA, IDC, in percentage, estimated

## Expanding OS Market Share Forecast for Japan



Source: Monthly Software Industry Report

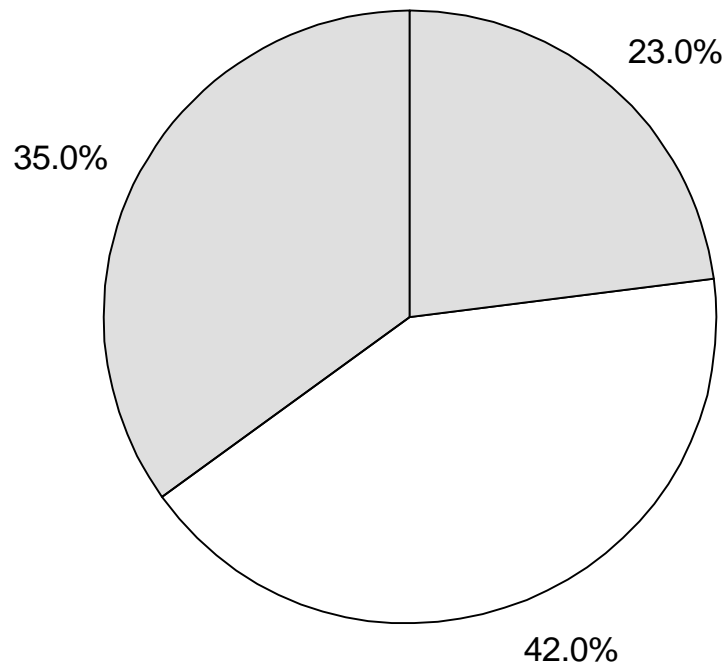
## WKS Shipment in Japan 1988 - 1994



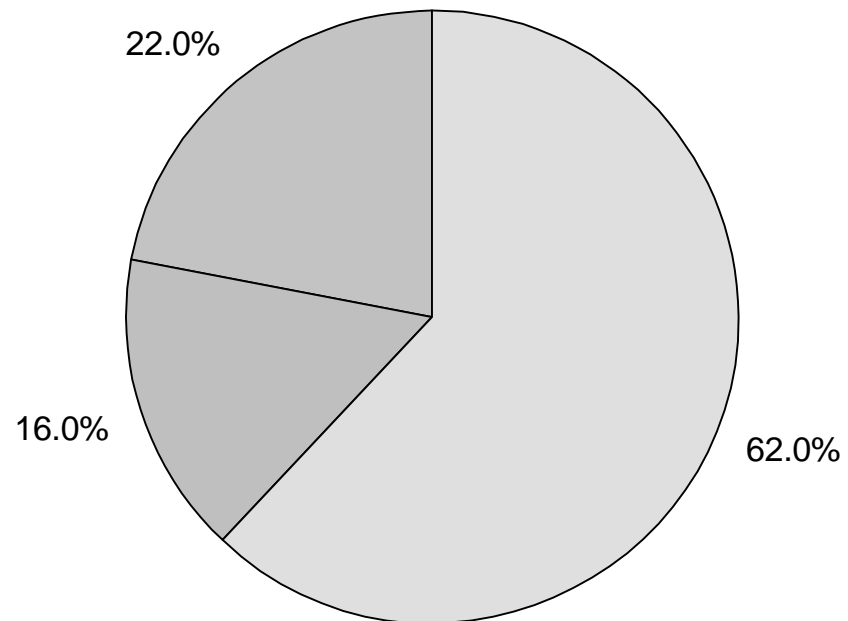
1988	1989	1990	1991	1992	1993	1994
30.7	47.5	75.8	160.0	220.0	280.0	360.0

Source: various, per 1000 Units; forecast

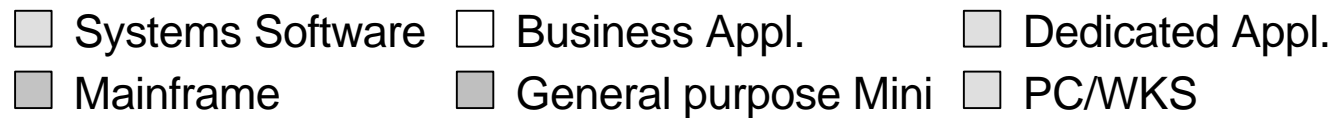
# Software Products & Hardware Platforms in Japan



SW

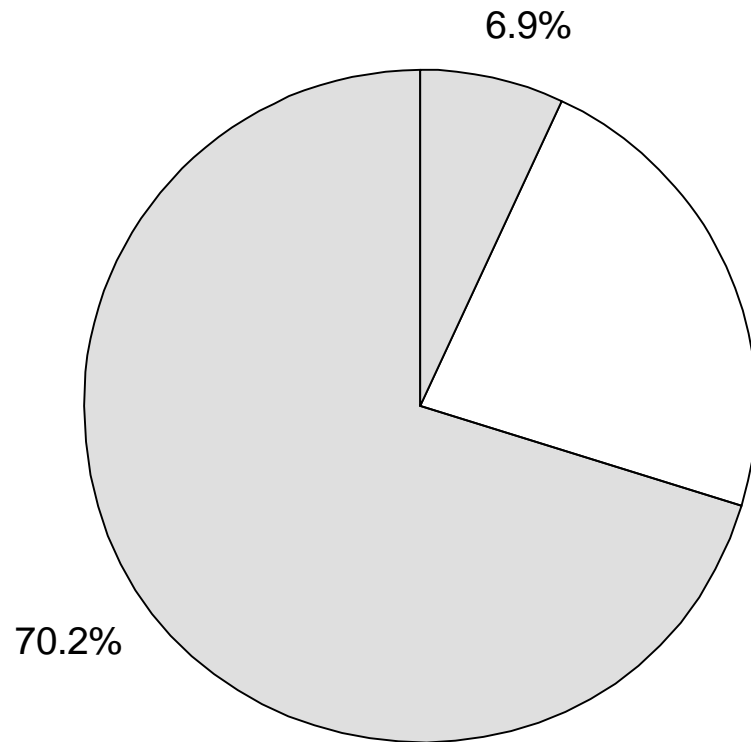


HW

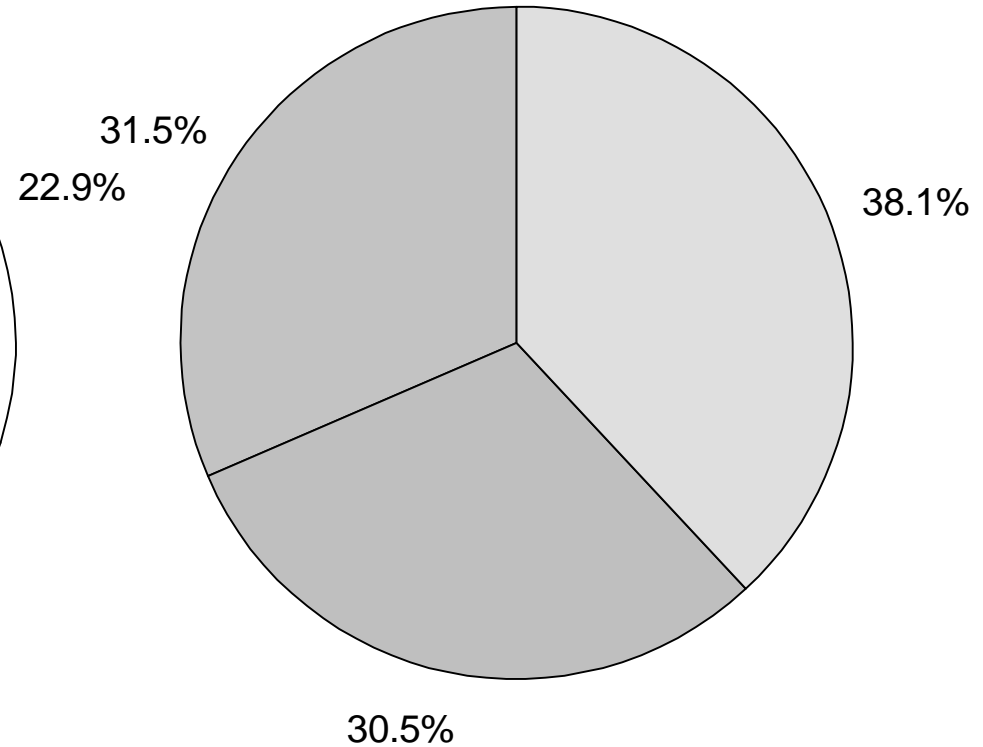


Sources: JISA, SOFTIC, in percentage

# Interest in foreign SW on HW Platforms



Foreign SW

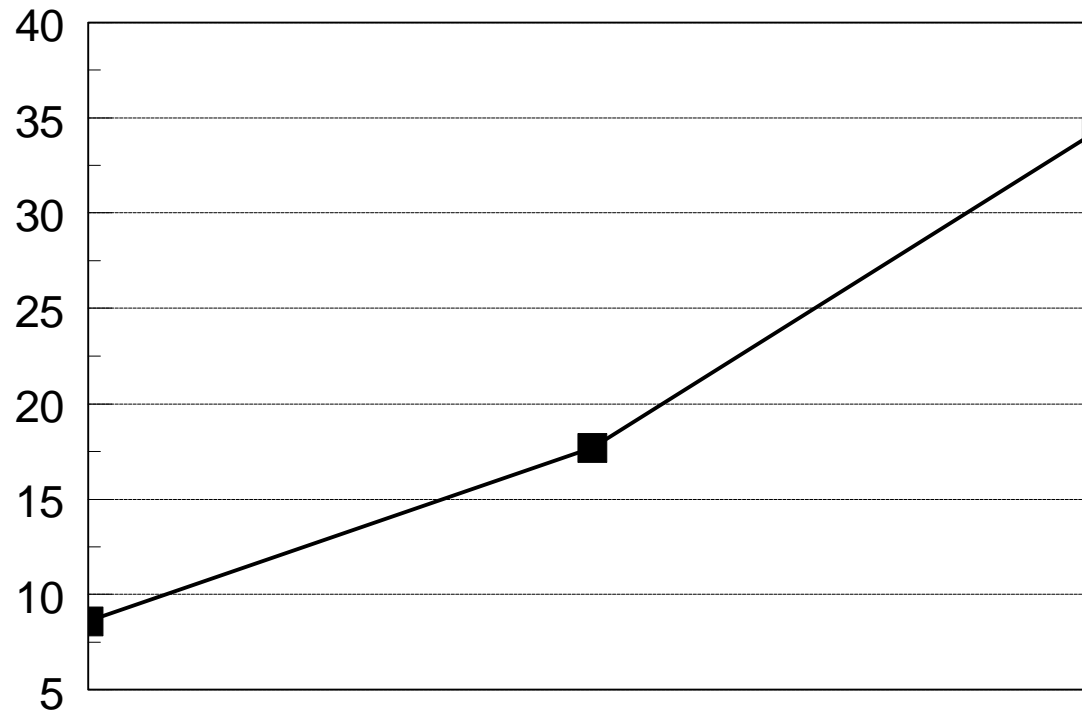


HW Platform

Legend:   
■ Yes   
□ NO   
■ no Answer   
■ UNIX   
■ MAC   
■ IBM

Sources: JPL

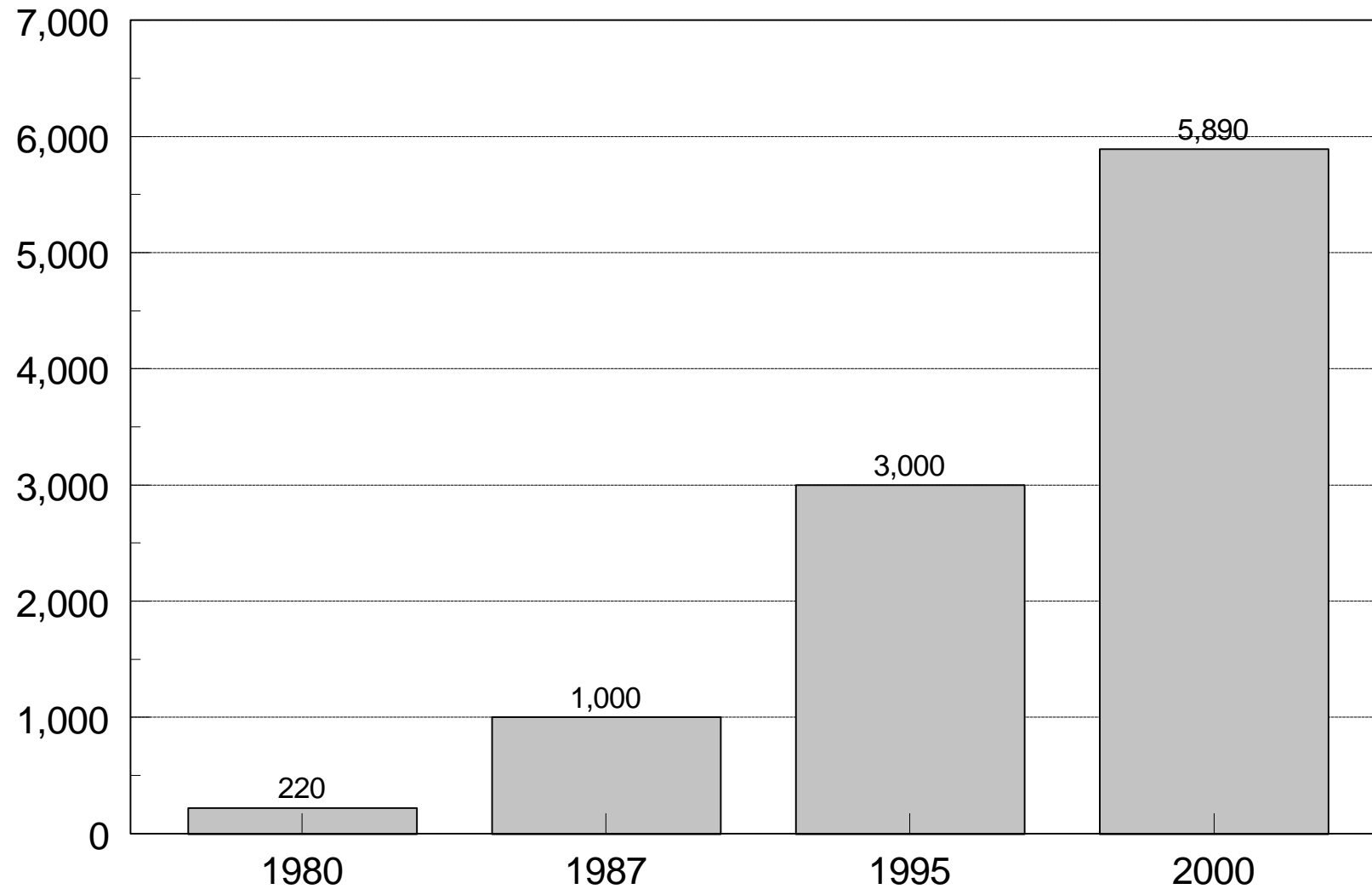
## Software Demand Forecast 1990's



1990	1995	2000
8.6	17.7	34.3

Source: JQL (MITI Survey), Trillions of \

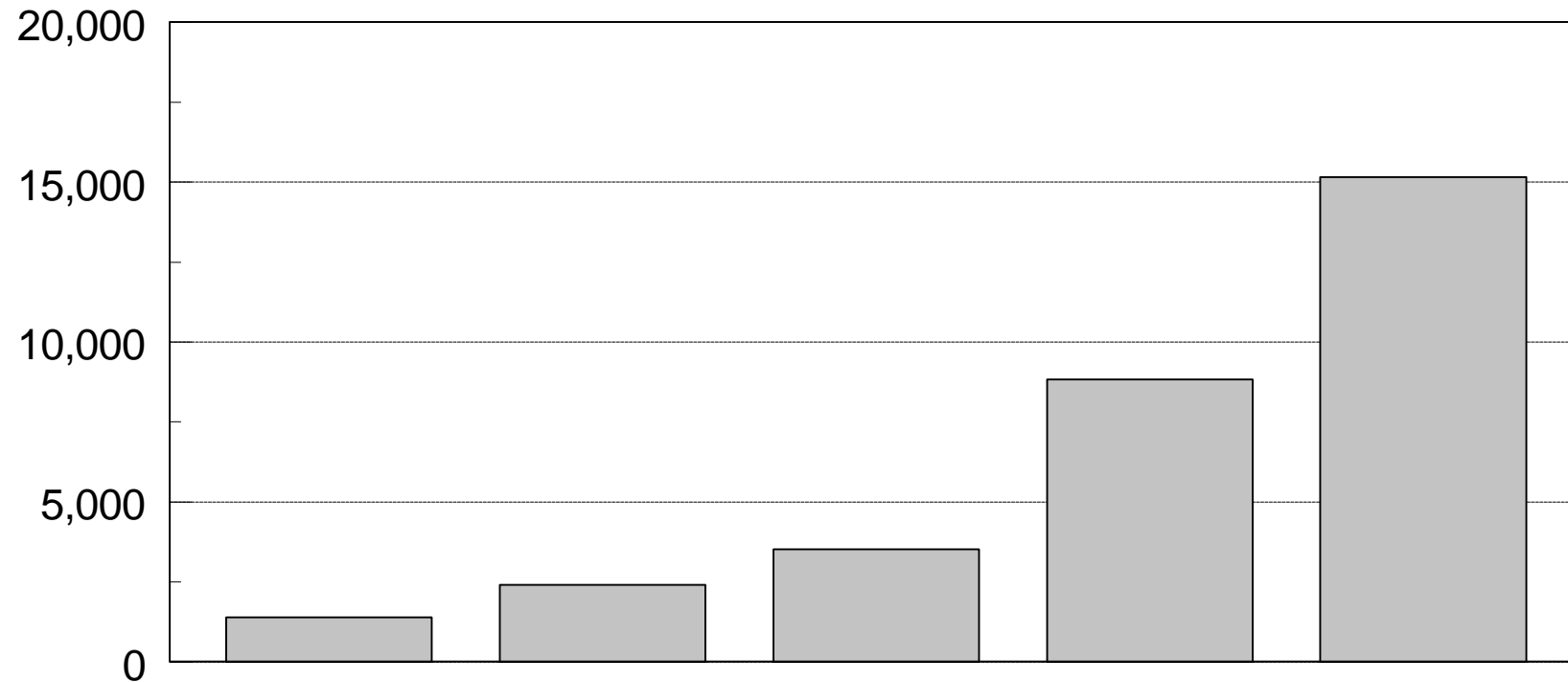
## Projection of Computer Software Market



Source: Jetro, Units : Billion \,estimated



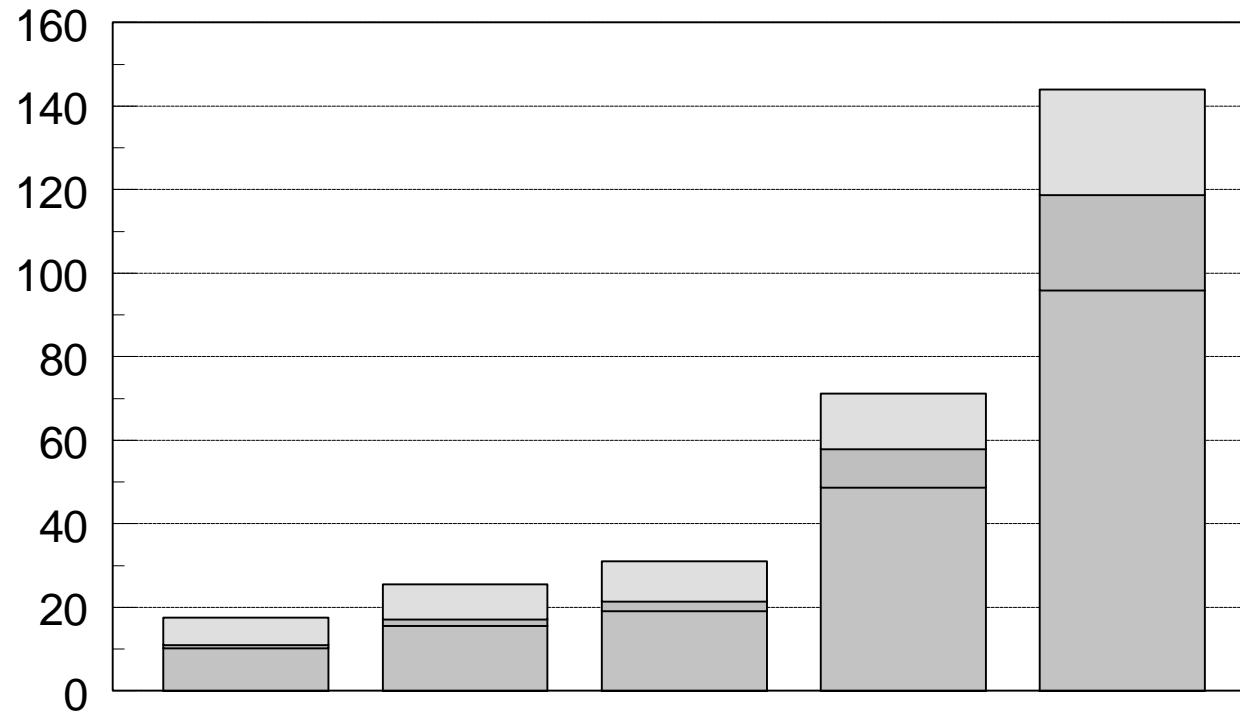
## Software Products Trends in Japan






1987	1988	1989	1995	2000
1,406	2,421	3,532	8,841	15,158

Sources: JISA, in 100 million \, forecast

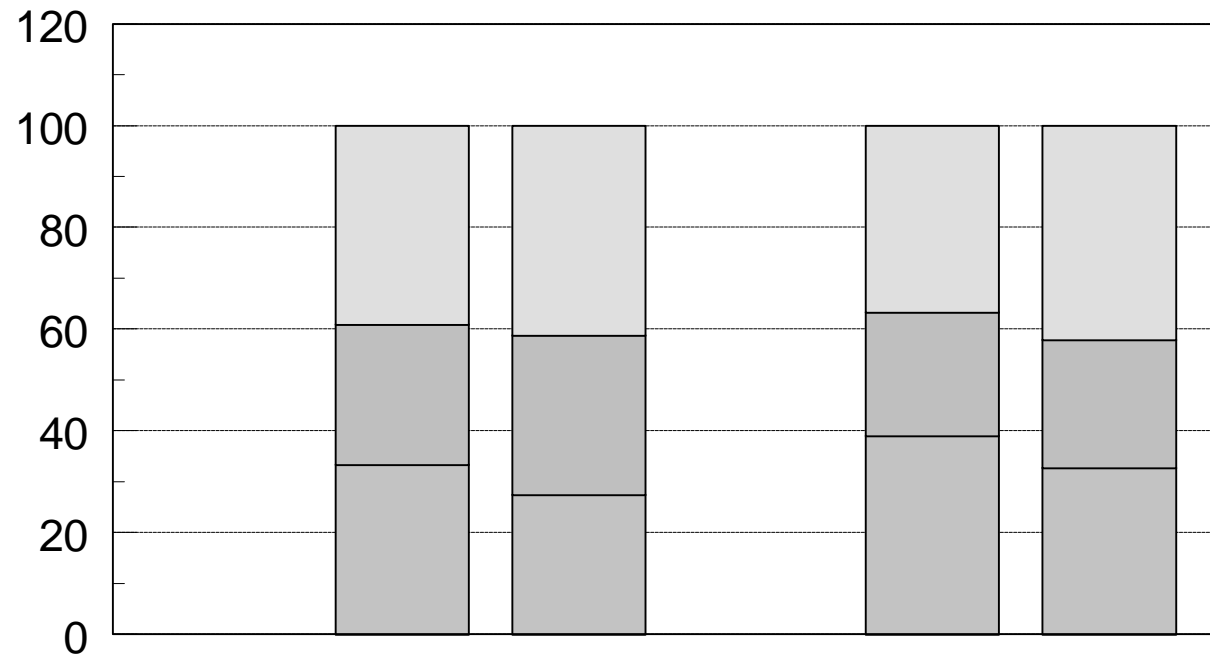
## Type of Software Product Development






	1987	1988	1989	1992	1994
Independently 	10.3	15.7	19.3	48.9	96.0
Developer 	0.8	1.6	2.3	9.1	22.9
Imported 	6.5	8.2	9.5	13.2	25.1

Sources: JISA, in 100 million \

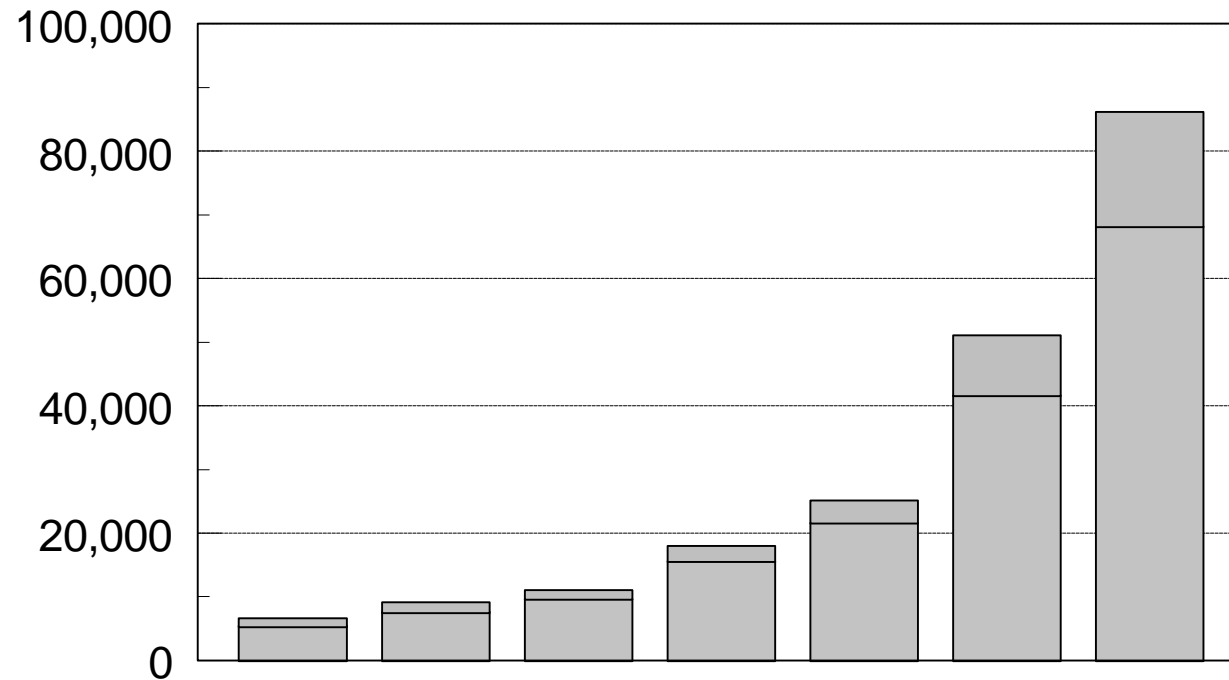
## Comparison of Software Products Market USA vs. Japan (1990 and 1995)





	USA	1990	1995	Japan	1990	1995
System/Utilities 		33.4	27.5		39.1	32.8
Application Tools 		27.6	31.3		24.3	25.2
Application Solution 		39.0	41.2		36.6	42.0

Source : IDC 1991, in percentage; forecast

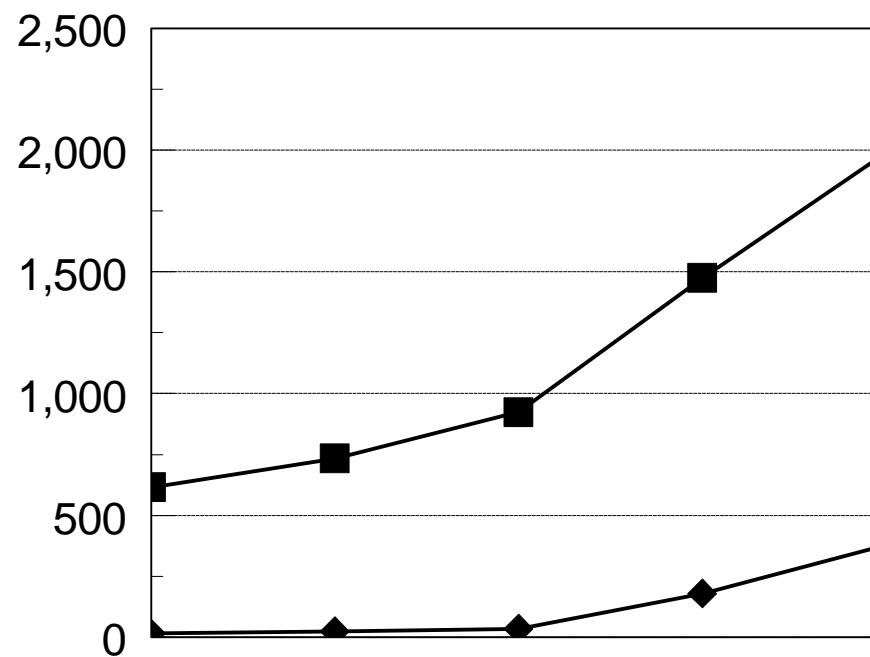
## Software development & Programming



	1985	1986	1987	1988	1989	1995	2000
Custom Software 	5,360	7,567	9,639	15,571	21,593	41,585	68,137
Software Products 	1,220	1,560	1,406	2,421	3,532	9,520	18,100

Sources: JISA, MITI, Units : 100 Million \\  
estimated figures

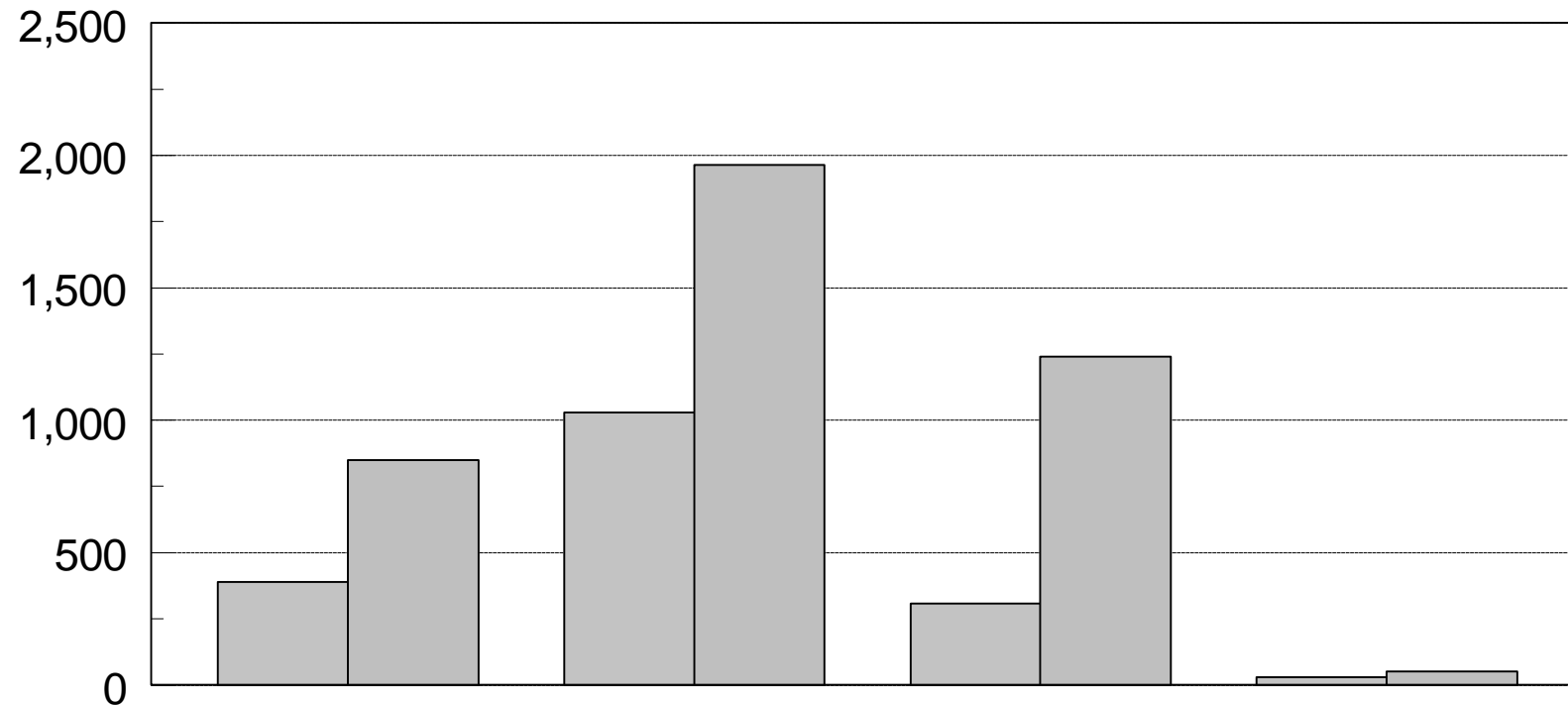
## Custom Software & Software Products in Software Development Sales



	1987	1988	1989	1992	1994
Custom Software <span>■</span>	615.5	733.9	925.1	1,475.2	1,989.5
Software Products <span>◆</span>	14.3	24.4	33.9	179.2	377.6

Sources: JISA, in 100 million \; forecast

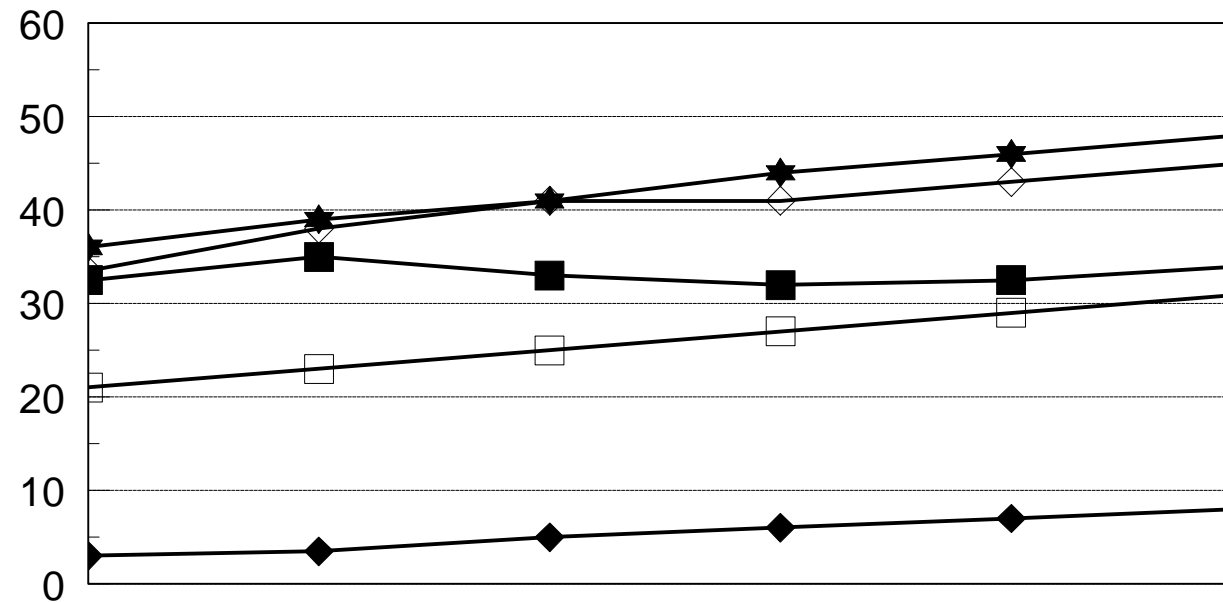
## Software Service Market Forecast for 1993








	Packaged Software	Coustom Software	System Integration	Consulting
1988	390	1,030	310	30
1993	850	1,965	1,242	53

Source: IDC, in Billions of \; forecast

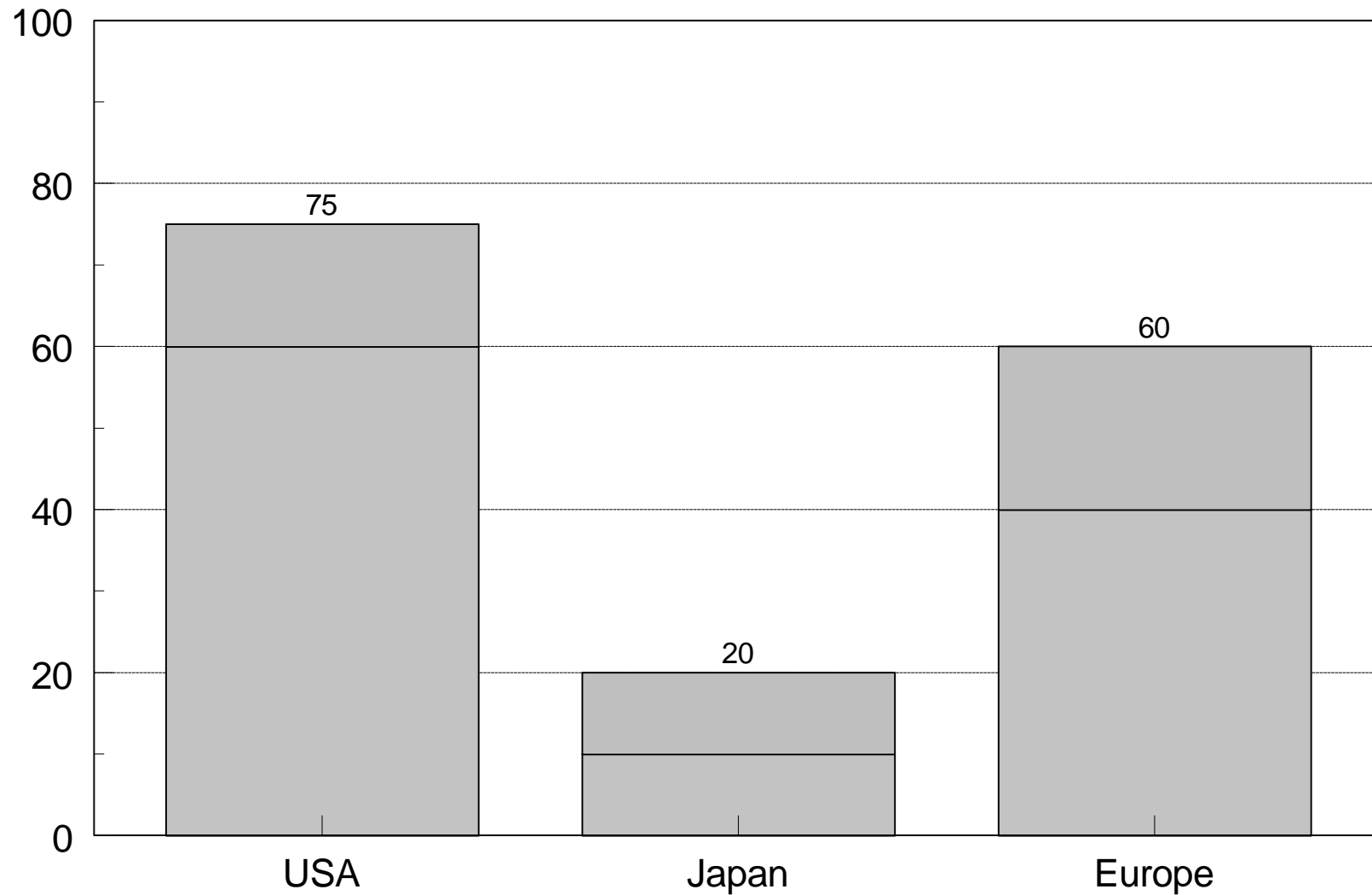
## Market share of packaged software 1983 - 1988



	1983	1984	1985	1986	1987	1988
USA 	32.5	35.0	33.0	32.0	32.5	34.0
Japan 	3.0	3.5	5.0	6.0	7.0	8.0
Germany 	36.0	39.0	41.0	44.0	46.0	48.0
France 	21.0	23.0	25.0	27.0	29.0	31.0
UK 	33.5	38.0	41.0	41.0	43.0	45.0

Source: IDC, 1988, in percentage

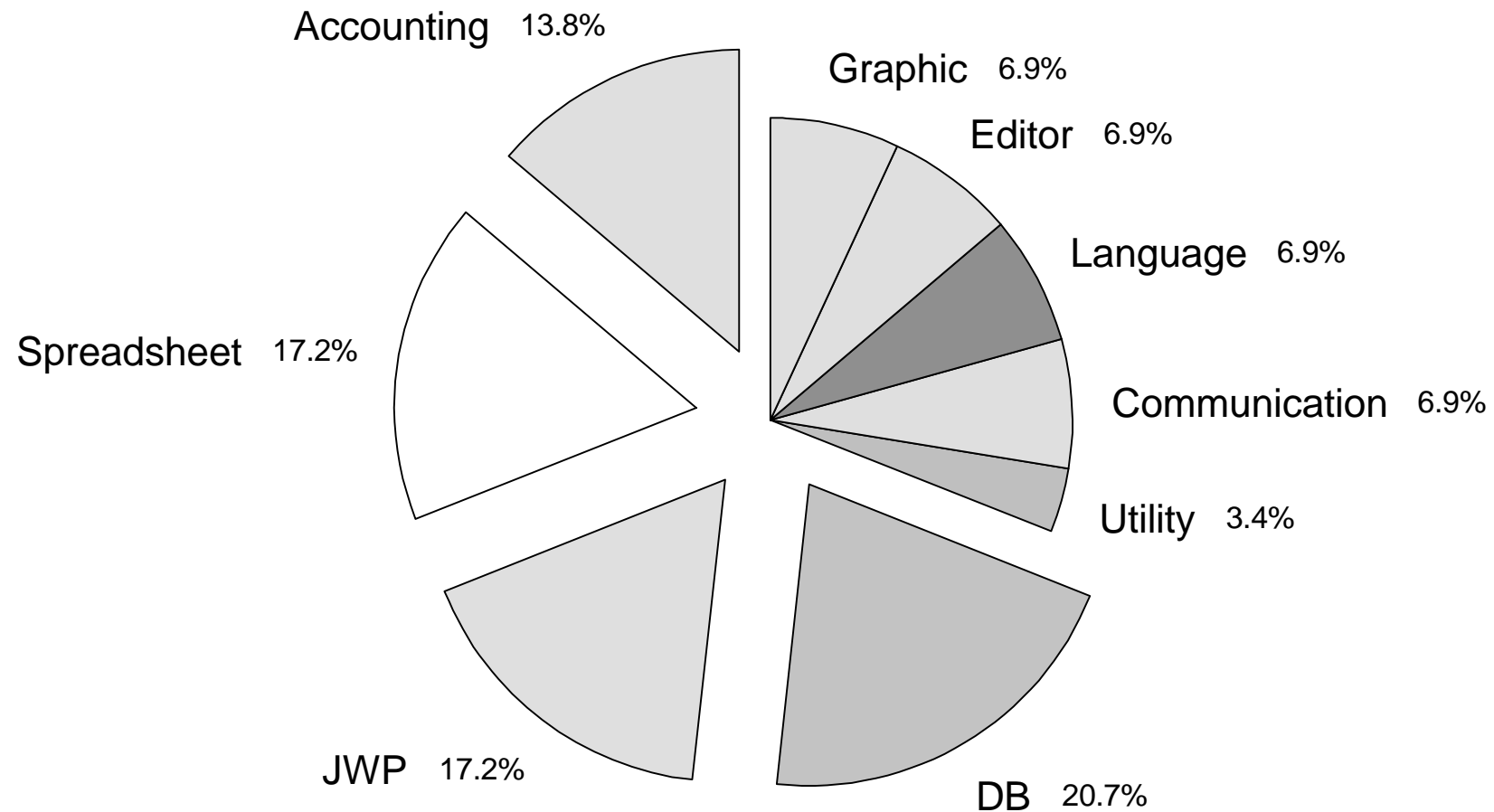
## Market share of packaged software



Source: various, in percent



## Top 30 Software Packages Sold in Japan



Sources: IDC 1990, in percentage

## Chapter 3

# Doing Business in Japan

In this section I want to give a short impression about the way of doing business in Japan. This "modus operandi" is different from the most western countries. In Japan a long term relationship between business partners is much more common than short term deals. Japanese companies show a strong tendency to choose a business partner carefully.

If you are a foreign software vendor and you want to penetrate the Japanese market, it is not enough to have a good product. Furthermore you have to do business in Japanese style. Depending on the way you choose to enter the Japanese market (branch/subsidiary, partnership, distributor, licensing, joint venture, ...) you have to get in touch with Japanese company executives or government officials. To get this smoothly settled you should know at least the basic Japanese business etiquette. In Japan it is much more common, than in the US or Europe, that sales representatives visit there (also future) customers. It is very important to do business face to face instead via telephone, fax or mail order. If you do not speak Japanese you should always take an interpreter with you to the customer. As meetings are very important in Japan, it takes long to get to a final decision, so you have to be patient. The Japanese companies are seeking long term business partners, so those if they have once chosen a partner they consider this relationship to last a long time. If you reached this stage you can be sure that you found a long term, highly committed business partner.

I cannot describe all the different business etiquette that are social rituals in Japan. At least I will try to give you a short impression about the complex system of business

etiquette in Japan.

Before the start of a meeting it is tradition to exchange business cards (*meishi's*). Be sure to have a set of bilingual business cards. English on one side and Japanese on the other side. Even if your business partners speak perfectly English it will show them that you are willing to respect the cultural difference to Japan. An other reason for business cards in Japan is that the Japanese judge a person from their social rank and the working place. Sometimes the cards used to remember the name and position of the other persons by placing the cards in seating order in front of them on the table. To show your commitment for this business bring people of rank with you to the negotiations. Even if they have nothing to do with this specific deal it will show your future Japanese business partner that this meeting is important for you. A good idea is also to present some gifts after successful negotiations, like specialities of your country or illustrated books about your country. Before the meetings you should send an agenda or other information to your business partner (e.g., in Japanese) so that they have time to study this material and get prepared for the meeting. If you do a presentation bring enough hand outs with you for everybody who participates in this meeting. Do not try to force somebody to finish the deal fast. You cannot expect to cut a deal within a couple of days. The base of the Japanese system of decision making is the consensus of all people who are involved in that project. If you receive an invitation to go out or play golf you should accept it because the Japanese executives do much of their business during this kind of entertainment. If you aim for a big deal it is not a bad idea to invite the main decision makers for a dinner, golf or even a trip to see your main office and to talk with the company heads. You should always show a great commitment to your Japanese customers. They should feel that you are not out for the quick dollar, that you also want to establish a long term relationship between the companies.

If you have done the negotiation and if both sides settled an agreement you should start to get the paperwork done. A lawyer in your business delegation will give the whole discussion a very negative aspect. For the Japanese delegation it would look like that you do not trust them. After the business partners have settled an agreement the lawyers should start their work.

If you deal with smaller products it is also important to have, e.g., an active sales force or distributor, which establishes and maintains contacts with (prospective) customers.

Besides that he should provide the necessary service and support to your customers. If you work with, e.g., a distributor you should visit Japan every couple of month to talk with them, provide the latest information about your product and company policy. To make a good impression about your commitment for the Japanese market you should visit your main customers and listen to their problems and requests.

Besides the way of establishing a business relationship it should be considered that in Japan a customer is treated like a God. The customer is the justification for the existence of the vendor. If the customer is not satisfied the vendor has failed in his duty. This reflects in a total commitment to service and quality. Service is one important duty of a vendor. If your customer has a problem the vendor should solve or at least help the customer to solve this problem. Despite the amount of money or man power is spent. The customer expects this kind of service. Also the warranty periods are longer then usual in the US or Europe. This is one reason that certain products are more expensive than their counterparts in the US or Europe. Because they include hidden charges for the expensive and long term after sales service.

Quality is one of the highest demands in Japan. Acceptable or good quality is not good enough. A product should always meet the highest possible quality requests. A breakdown is considered as unthinkable and if it happens the problem should be solved in the shortest possible time. Even if the customer has only a small problem someone from the service personnel of the vendor should be able to provide a solution for the customer's problem. If the unthinkable breakdown happens the vendor should be able to help the customer to prevent a disaster and the service personnel should, fastest as possible, repair or fix the problem.

In Japan the demands of custom build software is very high. You can expect requests from your customers to customize software on their demand. If your software package is programmable you can expect calls from your customers who are expecting help and advice in programming or adapting the software to their business environment. If you start doing business in Japan you have to be aware of the different business environment and social cultural rituals. Service, Quality and a high commitment for the customer's needs will help you to succeed in the Japanese marketplace.

# Chapter 4

## Cultural Differences

In this chapter I will give an introduction about the efforts, that have to be made to localize, in this case to japanize, software. I will show the main differences and in the next chapters I will introduce the reader to possible solutions (starting from 163).

### 4.1 Introduction

The general difference between the European/American culture and the Japanese culture is not only that Japan is an Asian country. It is, based on the different culture and religions, also the way they do the things the do. This is affected not only by the Japanese ”modus operandi”, it is also effected by the environment, which is given by their tradition, language and writing. Some people refer to this as ”Japanese language problems” (see [31]), but I think that only we got the problems and not the Japanese. That means that we have to modify our software if we want to sell it on the Japanese market.

So let us have a closer look on the requirements for the localization / internationalization of a software product for the Japanese market (see [2, 1]) :

- Numbers, the representation of numbers, decimal separator, . . .
- Currency, representation of the currency, average number of digits, decimal separator.

- Date convention, which date convention is used. How is a date represented? Are there other calendars then the Gregorian calendar used.
- Character set, the use of alphabet & numerals (the roman alphabet is called Romaji), Hiragana, Katakana and Kanji; special characters, double & single byte character sets. Special requirements like :
  - input methods (e.g., Keyboard, Pen) for Hiragana, Katakana and Kanji.
  - storing of characters as 1 or 2 Byte code (SBCS, DBCS)
  - displaying and printing characters
  - different standards for representation of character sets
- paper size
- units, etc.
- Manual, On-line help, support and telephone hotline should be in Japanese, e.g., a local office in Japan
- programming language (application) should be able to handle the Japanese character set
- To provide the right service for Japanese customers

In order to do this you need a knowledge about the cultural specialities which apply in the Japanese language environment. I will give you an overview on the next couple of pages.

## 4.2 Japanese Character Sets

When the Japanese started to build the first computer systems back in the 1960s they followed the example given by the US computer industry (see [3]). This first computer types were built like US models and used the same type of character set called ASCII (see figure 4.1, page 65). It was, for the stage of technology at this time, quite difficult to handle the specific Japanese character sets like the syllable alphabets Hiragana and Katakana or the thousands of ideographic Chinese characters called Kanji. This type of computer systems were not very sophisticated for the Japanese computer user. As mentioned before it is possible to write Japanese names and addresses in Romaji but it makes it very difficult to read a Japanese text, written entirely in Romaji. The Japanese use Kanji characters which are ideographic characters. This means that Kanji characters (or a combination of them) represent an idea, meaning or thought. It would be possible to write down the pronunciation but this is quite difficult because there are different systems to transliterate a Japanese Yomi (pronunciation) to the roman (Romaji) characters (e.g., Hepburn, Nipponsiki; see figure 4.2 starting from page 69). A Kanji character could have so different spellings depending on chosen transliterating system. Besides that a Kanji character could have different Yomi's (see c in figure 4.38 on page 125) or a Yomi could have different Kanji characters (see b, meiji, in figure 4.38 on page 125) depending on the meaning. This makes it very difficult for Japanese to use only Romaji.

### 4.2.1 ASCII and Katakana

The next step of the Japanese computer industry was to adapt the Katakana alphabet to the computer character set. This had some reasons, like :

- only a limited number of characters
- useful to express foreign and Japanese " words "
- relatively easy to implement on a computer
- depending on their shape they are easy to print or display

- does not require a Font End Processor (see page 115) for the input of the characters. It is easy to realize with a new level of characters on a standard keyboard (e.g., see figure 4.7 on page 74, [28]).

To implement Katakana characters was the easiest way to add, at least some, Japanese capabilities to a computer system. It has only a limited number of characters (which where fitting in the space above 127 decimal,  $7F_{Hex}$ ) so that it was possible to use the 7(8)-bit architecture. You could get along without major changes (see figure 4.9 and 4.10 on page 76 and 77). This one byte code (Single Byte Character Set, SBCS) is called JIS X0201-1989 (the name changed, march 1987, from the old name JIS C6220-1976) and describes an enhanced ASCII character set which includes Katakana characters. The use of this Katakana character set has the advantage that, through the limited number of characters, it is possible to use a standard keyboard and shift between ASCII and Katakana input (see figure 4.7 and 4.8 on page 74 and 75, [28]). This keyboard layout is naturally defined by an own standard called JIS X6002-1984 (or the predecessor JIS C 6233-1980). On the keyboard layout appear the 52 small and capital Roman alphabet characters, ten numerals, 32 special characters (like !, \$, &, @, +, -, etc.), 8 Japanese special characters, 17 control characters (like CR, LF, ETX, DEL, ESC, ...) and 55 Katakana <sup>1</sup> characters. A standard definition does, unfortunately, not mean that everybody has to follow this definition. This causes that there are different keyboard layouts available.

## 7-bit JIS

This code exists in a 7-bit and a 8-bit version. The difference between the versions is that in the 7-bit (from 00 to  $7F_{Hex}$ ) version a Shift In (SI,  $0F_{Hex}$ , sometimes called Kanji In (KI)) and a Shift Out (SO,  $0E_{Hex}$ , sometimes called Kanji Out (KO)) character is used to shift between the ASCII and Katakana code table. This means that the system starts printing ASCII characters until it runs over a SO. All following characters are printed as Katakana characters. This stops when the system finds a SI, which switches back from Katakana mode to ASCII mode. The use of a SI and SO character to switch between code tables could cause some problems which I will later explain more in detail.

---

<sup>1</sup>Which is actually not the complete set of Katakana characters



### 8-bit JIS

This problem does not occur when your system is able to use the 8-bit version of the JIS X0201-1989. In this case the system must be able to work with 8-bit characters (called "8-bit clean", which was not always possible, e.g., in early implementations of UNIX, they sometimes used the highest bit as a parity bit). With the 8-bit version you do not have to use the SI and SO characters to switch between the ASCII and Katakana code table. The Katakana characters are just located in the, former unused, area above  $7F_{Hex}$ .

The use of this area could cause some problems when you work with, e.g., American IBM PC software. The IBM PC has a totally different codetable in the area between  $7F_{Hex}$  and  $FF_{Hex}$ . If you start using foreign software it could (or definitely will) happen that a screen mask looks very funny because instead of line-elements Katakana characters appear. This results in a nearly ASCII compatible character set. In the 7-bit version one of the code tables is nearly compatible to ASCII. In the 8-bit version the area below  $7F_{Hex}$  is nearly compatible. The only difference, which makes the character set just nearly compatible, is that the backslash ( $\backslash$ ,  $5C_{Hex}$ ) is replaced by the Yen symbol (see picture d in figure 4.32, page 114). The second replacement is the tilde ( $\sim$ ,  $7E_{Hex}$ ) which is replaced by the overline ( $\bar{\phantom{x}}$ ). All other characters correspond to their ASCII equivalent.

These Katakana characters have the same size as an ASCII character. The Katakana characters in this size are called half-width Katakana (in Japanese Hankaku). Still this was not a very sophisticated solution for the early Japanese computer users. The lack of Kanji characters was one of the important points which made them start to think about how to integrate Kanji characters into computer systems.

### 4.2.2 Development of Kanji Character Sets

To understand the following development of the Kanji character code set we have to have a look on non-electronic (see [8]) character sets which were used to define the JIS C 6226-1978. Which was leading towards JIS X0208-1990, the standard today.

The Japanese have about 40000 to 60000 "known" Kanji characters. The problem with that is that nobody is able to remember all of them. The Ministry of Education

started to restrict the number of Kanji characters for the use in education. Today a Japanese student learns about 2000 Kanji characters.

The historical development of the standard was started with the table of sanctioned Kanji characters for education. This first table was called Toyo Kanji and contained, in 1946, 1850 Kanji characters. This table was replaced by the Joyo Kanji character table in 1981. This table contains now 1945 Kanji character (see figure 4.11, page 78). The other tables which were used to form the standard character set are the Gakushu Kanji (replaces the older Kyoiku Kanji table with 881 Kanji characters) with 1006 characters (increased in 1992 from 996 Kanji characters) and the Jinmei-yo Kanji character table which has since 1990 284 characters (increase from 85 characters in 1946 to 112 characters in 1976 then to 166 characters in 1981). An interesting fact is that Gakushu Kanji is a subset of Joyo Kanji (see [8]).

### **Double Byte Character Sets**

This non-electronic character sets were used to define the actual DBCS character set standard JIS X0208-1990. Besides the Kanji, Hiragana (83) and Katakana (86) characters the standard includes alphanumeric characters (10 numerals, 52 Roman characters), special characters (147 symbols), Greek (48) and Russian (66) letters and rule line elements (32). During the years there were some changes (X208 was first established 1978, the first change occurred 1983, the actual version is from 1990) which added some new Kanji characters, changed the shape of some characters or a change in the positions of some characters has taken place. Today this standard contains two levels with 2965 characters in level 1 and 3388 characters in level 2. In 1990 the JSA introduced a supplementary DBCS character set which is called JIS X0212-1990 (sometimes referred as JIS level 3) with additional 6067 characters. In addition to 5801 Kanji characters this standard contains 21 special characters and 245 Latin (Roman), Cyrillic and Greek characters (mostly with diacritical marks, characters like, e.g., German Umlauts, the French, Spanish or Danish special characters). This leaves us with a total number of 12156 standard characters, divided into three levels. Regarding the fact that JIS X0212-1990 is a very young standard the most systems use only the characters defined by the JIS X0208-19XX standard (see [25], [26], [27]). Nevertheless that this tremendous number of characters needs a lot memory. It is also impossible to represent this characters by using a SBCS. In order to represent this

huge number of characters we need at least a Double Byte Character Set (DBCS).

In a standard 7 ( or 8) bit environment we could use a character set which contains 127 (or 255) characters. This is enough to carry a standard ASCII character set and some national extensions, but it is not big enough to handle thousands of ideographic Kanji characters. In order to handle the tremendous number of characters we have to extend the number of bits which hold the character information. In a 7-bit environment a logical step is to use two 7-bit bytes (14-bits) to hold the information this would give us the possibility to store up to  $2^{14}$  (16384) characters. If we use two 8-bit bytes we are able to store up to  $2^{16}$  (65536) characters. The arising problem is how to distinguish between SBCS characters and DBCS characters. In order to stay compatible with the old SBCS character set you have to find a solution to determine if the actual byte is a SBCS character or belongs, as a part of it, to a DBCS character.

### Shifting between SBCS and DBCS

Again, as mentioned above, it is possible to use the Shift In / Out mechanism to distinguish between SBCS and DBCS. This is quite useful in a 7-bit environment. Also it could be used in an 8-bit environment. Another possibility, in an 8-bit environment, is to use the MSB (Most Significant Bit) as a flag to show that this byte is a SBCS character (MSB = 0) or a part of a DBCS character (MSB = 1). A SBCS could look, in binary representation, like 0XXXXXXX and a DBCS would look like 1XXXXXXX 1XXXXXXX. Today the most large or medium sized systems uses a SI/SO (or KI/KO) sequence to switch between SBCS and DBCS characters. There is a recommendation from the JSA for this SI/SO sequence, but unfortunately the most hardware vendors have chosen different SI/SO sequences (usually between one and three bytes). Some examples for the SI/SO (KI/KO) sequences for New-JIS, Old-JIS, NEC-JIS, Lets-J, JBIS, JEF and IBM EBCDIC you will find in figure 4.1 (on page 66, [8], [9]).

Sometimes there are two different SI/KO sequences. One sequence switches back to the JIS-Roman character set. The other SO/KI sequence switches back to the ASCII character set.

Not only the SI/SO (KI/KO) sequences differ between the different implementations of a Kana/Kanji character set, also the location in the matrix which is defined by

the two bytes. Moreover some companies (like IBM) even do not use the JIS defined standard (for some examples see table 4.2 on page 67).

Starting from page 79 you will find the two byte matrices from different Kana/Kanji character sets. If you have a closer look on these matrices you will recognize that all vendors placed the JIS area or the extension area at different places. Although if the matrices are on the same place it does not mean that the same Kanji character appears at the same place. In the Japanese PC world the Shift JIS is the standard for the character set. This version of the JIS character set was moved to a different location (see figure 4.18 on page 85) because on this location it was possible to use the old 7-bit character set and the DBCS without a SI/SO (or KI/KO) sequence. In Shift JIS all 7-bit characters (SBCS) have the MSB set to 0 and look like 0XXXXXXX. If the MSB is set to 1 the byte is a part of a DBCS character (it looks like 1XXXXXXX 1XXXXXXX). An advantage of the Shift JIS design is that it is very easy to convert a JIS DBCS code to the corresponding Shift JIS DBCS code. To do this you could use the following formula ([7]) :

### Shift JIS and JIS

*SJIS is the two byte representation of the Shift JIS code and JIS the two bytes of the JIS code. SJIS<sub>1</sub> is the first byte and JIS<sub>2</sub> is the second byte of the code. The value for the bytes is between 00<sub>Hex</sub> and FF<sub>Hex</sub>*

```

SJIS1 = (JIS1 - 21Hex) / 2 + 81Hex
if SJIS1 ≥ 9FHex then JIS1 = JIS1 + 40Hex
if odd(JIS1) then begin
  SJIS2 = JIS2 - 21Hex + 40Hex
  if (SJIS2 ≥ 7FHex) then SJIS2 = SJIS2 + 1
end
else SJIS2 = JIS2 - 21Hex + 9FHex

```

The Shift JIS is mainly used in the PC world and in few workstations. The most vendors offer conversion routines between their own code set and JIS and Shift JIS.

Another fact about the different character sets is that the user defined characters are placed on different locations in the two byte matrices. The number of the, so

called Gaiji characters, differs in each of the vendor's implementations. These Gaiji characters are needed because some Japanese names are written with "nonstandard" Kanji characters. If an, e.g., insurance company wants to print an invoice with the name of the customer it is common practice to use a user defined Gaiji character for this purpose (when the name of the customer contains a Kanji character which is not available as a JIS standard character).

### 4.2.3 Japanese Character Set Mess

When Fujitsu started 1978 ([9]) to introduce their main frame implementation of Japanese language processing nearly every vendor has introduced a different DBCS character set (some even more than one DBCS, see table 4.3 on page 68). Some European companies used an even more outer space approach, by designing totally incompatible Japanese character sets. Actually does this policy not increase the chances for selling software.

Even if all of them support JIS or Shift-JIS via conversion routines none of the systems is compatible to an other system. This leaves us with some problems for the japanization of foreign software. For each hardware platform you have to check the implementation and make (even slight) changes to adapt to this vendors' platform and DBCS character set.

Column ----- Row	00	01	02	03	04	05	06	07
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	\$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

ASCII Alphanumeric (Part 1)

Figure 4.1:

Standard	Kanji In	Kanji Out (JIS-Roman)	Kanji Out (ASCII)
New-JIS (1983)	ESC \$ B	ESC ( J	ESC ( B
Old-JIS (1978)	ESC \$ @	ESC ( J	ESC ( B
NEC-JIS (1978)	ESC K	ESC H	n/a
Lets-J	93 <sub>Hex</sub> F0 <sub>Hex</sub>	n/a	93 <sub>Hex</sub> F1 <sub>Hex</sub>
JBIS	2B <sub>Hex</sub> (SOK)	n/a	2C <sub>Hex</sub> (EOK)
IBM EBCDIC DBCS	0F <sub>Hex</sub>	n/a	0E <sub>Hex</sub> †
JEF	28 <sub>Hex</sub>	n/a	29 <sub>Hex</sub> †
JIS X0201-1976	0E <sub>Hex</sub> SO	n/a	0F <sub>Hex</sub> SI
JIS X0202-1984	ESC ( I ‡		

Table 4.1: Different SO/KI and SI/KO codes

†= switches back to EBCDIC

‡= switches to half-width Katakana

Company	JIS	Extension
IBM (Host)	No	EBCDIC, IBM Basic Character set (3572 characters), IBM Extended Character set (3483 characters), IBM user (free) region (4370 characters)
IBM (PC)	Yes †	1880 User defined characters
AX Consortium	Yes †	448 User defined Hankaku characters, 752 User defined Zenkaku characters and 1024 Zenkaku characters defined in hardware
JEF (Fujitsu)	Yes	EBCDIC, 4039 Extended Kanji characters, 1083 Extended Non-Kanji characters, 3102 User defined characters
DEC	Yes	DEC JIS Extension up to 9621 User defined characters
LETS-J	Yes	Yes ‡
JBIS	Yes	Yes ‡
Hitachi	Yes	Yes ‡
NEC	Yes	Yes ddag

Table 4.2: Support of JIS standard character set

†= Shift JIS (see figure 4.18 on page 85)

‡= No information provided



Vendor	DBCS Implementation
Fujitsu	JEF
	JEF II
Hitachi	KEIS
NEC	JIPS
UNISYS	LETS-J
	JBIS
IBM	EGCS (old)
	DBCS (SAA)
PC-World	Shift-JIS
UNIX	EUC

Table 4.3: Different DBCS Implementations

Romaji ローマ字 the Hepburn system ヘボン式	Romaji ローマ字 Japanese system 日本式	Hiragana ひらがな	Katakana カタカナ
a	a	あ	ア
i	i	い	イ
u	u	う	ウ
e	e	え	エ
o	o	お	オ
ka	ka	か	カ
ki	ki	き	キ
ku	ku	く	ク
ke	ke	け	ケ
ko	ko	こ	コ
sa	sa	さ	サ
<u>shi</u>	<u>si</u>	し	シ
su	su	す	ス
se	se	せ	セ
so	so	そ	ソ
ta	ta	た	タ
<u>chi</u>	<u>ti</u>	ち	チ
<u>tsu</u>	<u>tu</u>	つ	ツ
te	te	て	テ
to	to	と	ト
na	na	な	ナ
ni	ni	に	ニ
nu	nu	ぬ	ヌ
ne	ne	ね	ネ
no	no	の	ノ

a)-1 Comparison of the Hepburn system and Japanese system

Romaji ローマ字 the Hepburn system ヘボン式	Romaji ローマ字 Japanese system 日本式	Hiragana ひらがな	Katakana カタカナ
ha	ha	は	ハ
hi	hi	ひ	ヒ
<u>fu</u>	<u>hu</u>	ふ	フ
he	he	へ	ヘ
ho	ho	ほ	ホ
ma	ma	ま	マ
mi	mi	み	ミ
mu	mu	む	ム
me	me	め	メ
mo	mo	も	モ
ya	ya	や	ヤ
yu	yu	ゆ	ユ
yo	yo	よ	ヨ
ra	ra	ら	ラ
ri	ri	り	リ
ru	ru	る	ル
re	re	れ	レ
ro	ro	ろ	ロ
wa	wa	わ	ワ
wo	wo	を	ヲ
n	n	ん	ン

a)-2 Comparison of the Hepburn system and Japanese system

Romaji ローマ字 the Hepburn system ヘボン式	Romaji ローマ字 Japanese system 日本式	Hiragana ひらがな	Katakana カタカナ
ga	ga	が	ガ
gi	gi	ぎ	ギ
gu	gu	ぐ	グ
ge	ge	げ	ゲ
go	go	ご	ゴ
za	za	ざ	ザ
<u>ji</u>	<u>zi</u>	じ	ジ
zu	zu	ず	ズ
ze	ze	ぜ	ゼ
zo	zo	ぞ	ゾ
da	da	だ	ダ
<u>ji</u>	<u>di</u>	ぢ	ヂ
<u>zu</u>	<u>du</u>	づ	ヅ
de	de	で	デ
do	do	ど	ド
ba	ba	ば	バ
bi	bi	び	ビ
bu	bu	ぶ	ブ
be	be	べ	ベ
bo	bo	ぼ	ボ
pa	pa	ぱ	パ
pi	pi	ぴ	ピ
pu	pu	ぷ	プ
pe	pe	ぺ	ペ
po	po	ぽ	ポ

a)-3 Comparison of the Hepburn system and Japanese system

Romaji ローマ字 the Hepburn system ヘボン式	Romaji ローマ字 Japanese system 日本式	Hiragana ひらがな	Katakana カタカナ
kya	kya	きゃ	キャ
kyu	kyu	きゅ	キュ
kyo	kyo	きょ	キョ
kwa	kwa	くわ	クワ
<u>sha</u>	<u>sy</u> a	しゃ	シャ
<u>shu</u>	<u>sy</u> u	しゅ	シュ
<u>sho</u>	<u>sy</u> o	しょ	ショ
<u>cha</u>	<u>ty</u> a	ちゃ	チャ
<u>chu</u>	<u>ty</u> u	ちゅ	チュ
<u>cho</u>	<u>ty</u> o	ちょ	チョ
nya	nya	にゃ	ニャ
nyu	nyu	にゅ	ニュ
nyo	nyo	にょ	ニョ
fa	fa	ふぁ	ファ
fi	fi	ふぃ	フィ
fe	fe	ふぇ	フェ
fo	fo	ふぉ	フォ
hya	hya	ひゃ	ヒャ
hyu	hyu	ひゅ	ヒュ
hyo	hyo	ひょ	ヒョ
mya	mya	みゃ	ミャ
myu	myu	みゅ	ミュ
myo	myo	みょ	ミョ
rya	rya	りゃ	リャ
ryu	ryu	りゅ	リュ
ryo	ryo	りょ	リョ

a)-4 Comparison of the Hepburn system and Japanese system

Romaji ローマ字 the Hepburn system ヘボン式	Romaji ローマ字 Japanese system 日本式	Hiragana ひらがな	Katakana カタカナ
gya	gya	ぎゃ	ギャ
gyu	gyu	ぎゅ	ギュ
gyo	gyo	ぎょ	ギョ
gwa	gwa	ぐわ	グワ
<u>ja</u>	<u>zya</u>	じゃ	ジャ
<u>ju</u>	<u>zyu</u>	じゅ	ジュ
<u>je</u>	<u>zye</u>	じえ	ジェ
<u>jo</u>	<u>zyo</u>	じょ	ジョ
<u>ja</u>	<u>dya</u>	ぢゃ	ヂャ
<u>ju</u>	<u>dyu</u>	ぢゅ	ヂュ
<u>jo</u>	<u>dyo</u>	ぢょ	ヂョ
bya	bya	びゃ	ビャ
byu	byu	びゅ	ビュ
byo	byo	びょ	ビョ
pya	pya	ぴゃ	ピャ
pyu	pyu	ぴゅ	ピュ
pyo	pyo	ぴょ	ピョ
va	va	-	ヴァ
vi	vi	-	ヴィ
vu	vu	-	ヴ
ve	ve	-	ヴェ
vo	vo	-	ヴォ
dya	dya	でい	デイ
tya	tya	てい	テイ

a)-5 Comparison of the Hepburn system and Japanese system

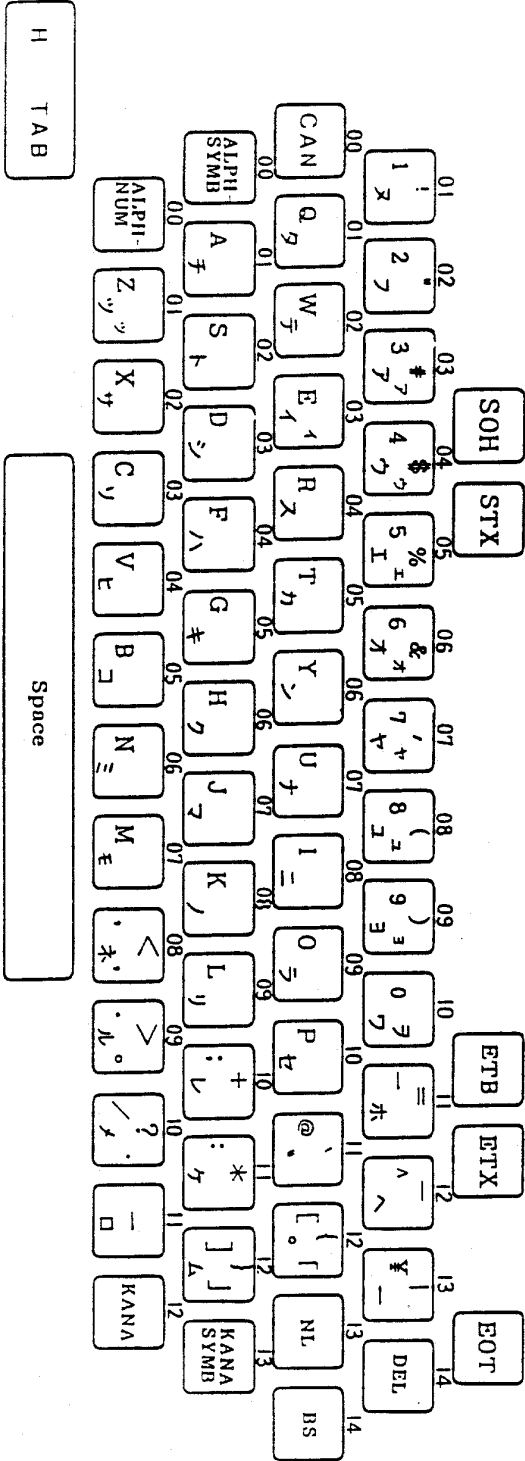


Figure 4.7: Katakana Keyboard, Page 1

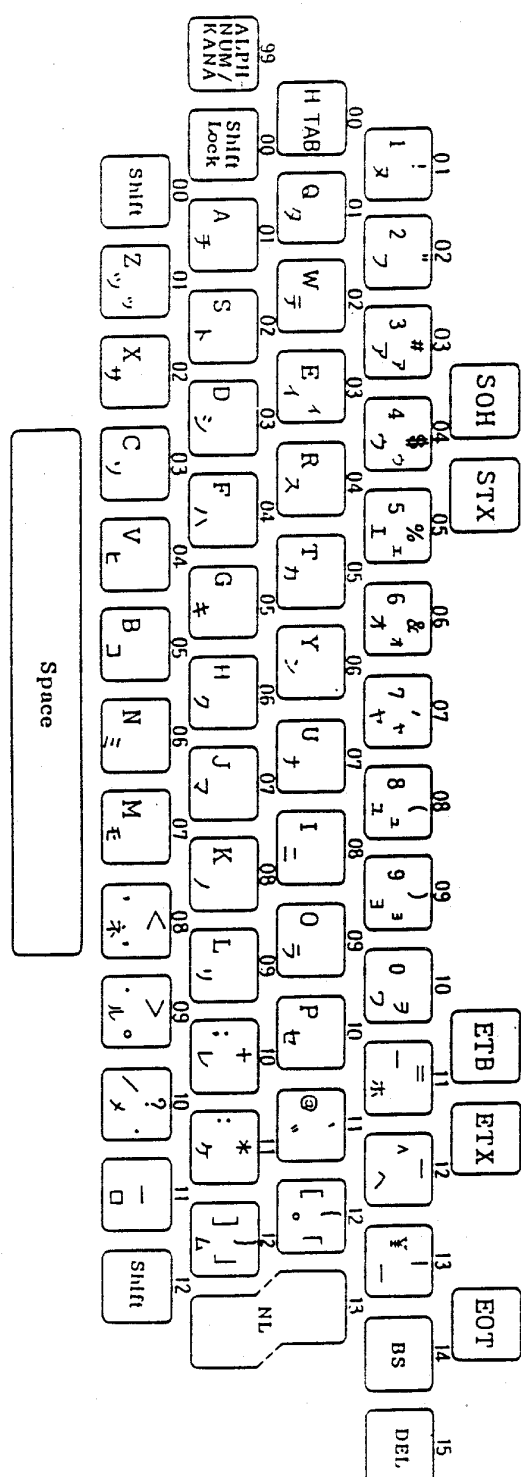


Figure 4.8: Katakana Keyboard, Page 2

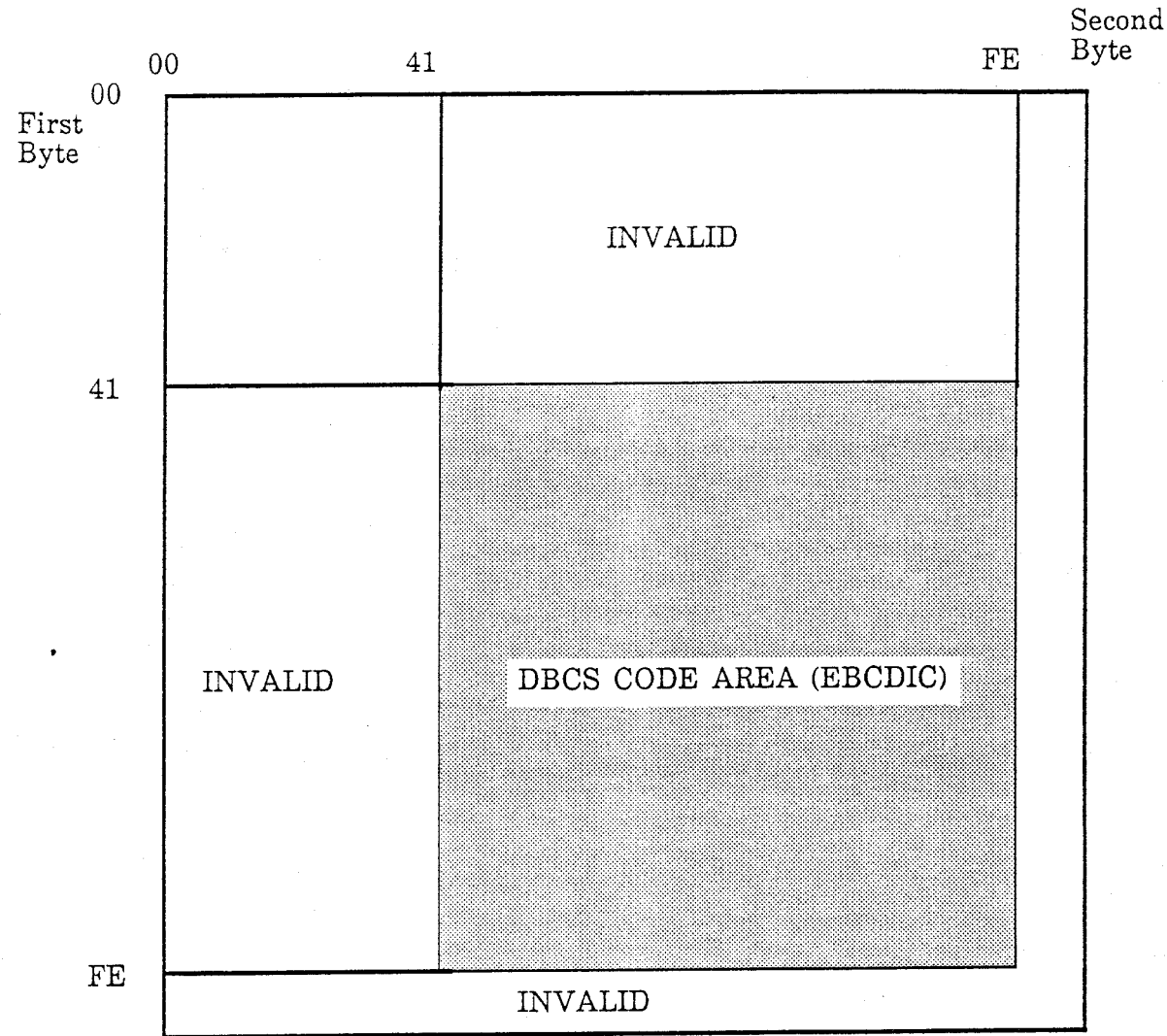


Column ----- Row	00	01	02	03	04	05	06	07
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	\$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	¥	l	
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	-
F	SI	US	/	?	O	—	o	DEL

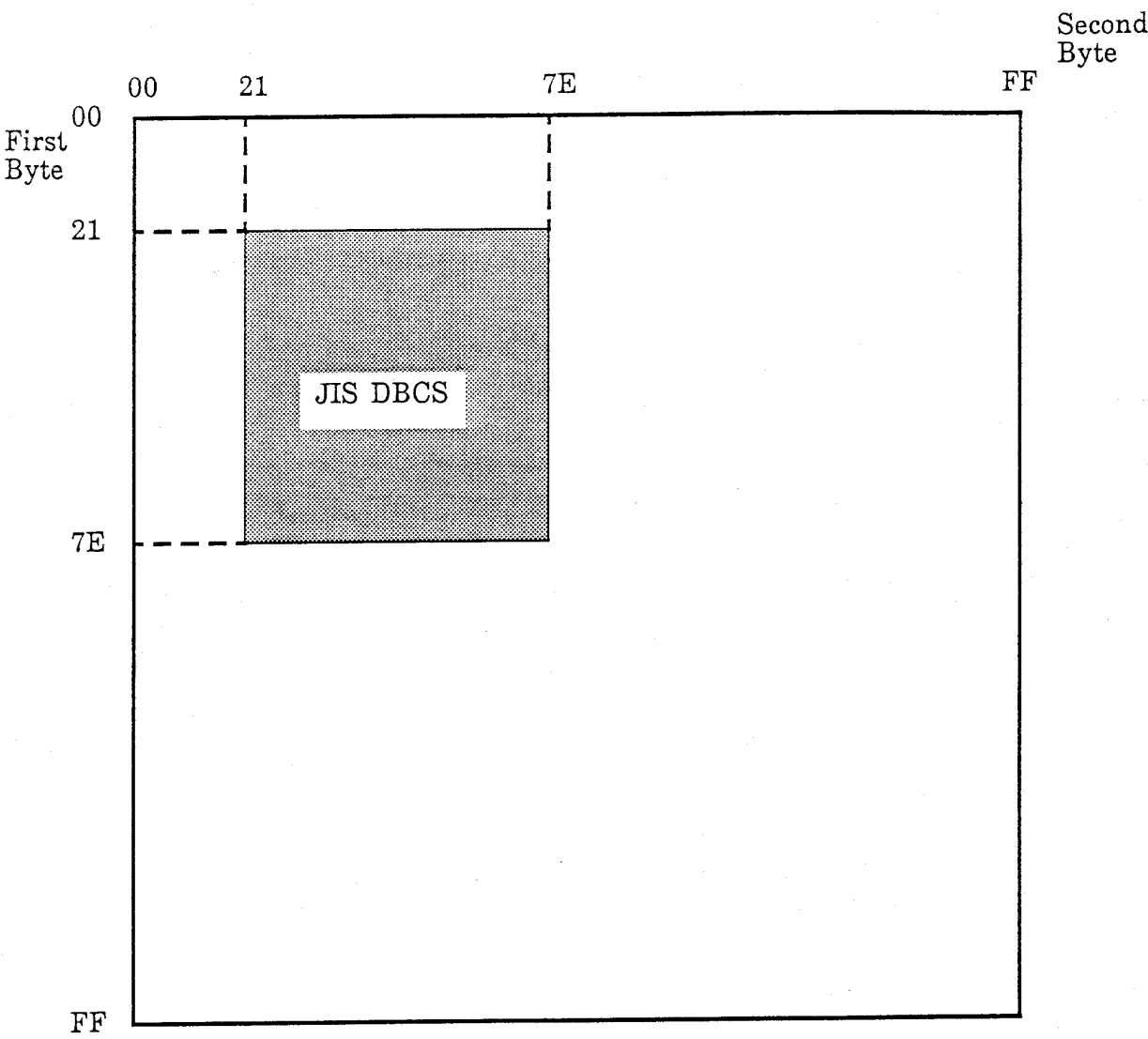
Column ----- Row	08	09	0A	0B	0C	0D	0E	0F
0	■	■	■	-	タ	ミ	■	■
1	■	■	。	ア	チ	ム	■	■
2	■	■	「	イ	ツ	メ	■	■
3	■	■	」	ウ	テ	モ	■	■
4	■	■	、	エ	ト	ヤ	■	■
5	■	■	・	オ	ナ	ユ	■	■
6	■	■	ヲ	カ	ニ	ヨ	■	■
7	■	■	ア	キ	ヌ	ラ	■	■
8	■	■	イ	ク	ネ	リ	■	■
9	■	■	ウ	ケ	ノ	ル	■	■
A	■	■	エ	コ	ハ	レ	■	■
B	■	■	オ	サ	ヒ	ロ	■	■
C	■	■	ヤ	シ	フ	ワ	■	■
D	■	■	ユ	ス	ヘ	ン	■	■
E	■	■	ヨ	セ	ホ	〃	■	■
F	■	■	ツ	ソ	マ	。	■	■

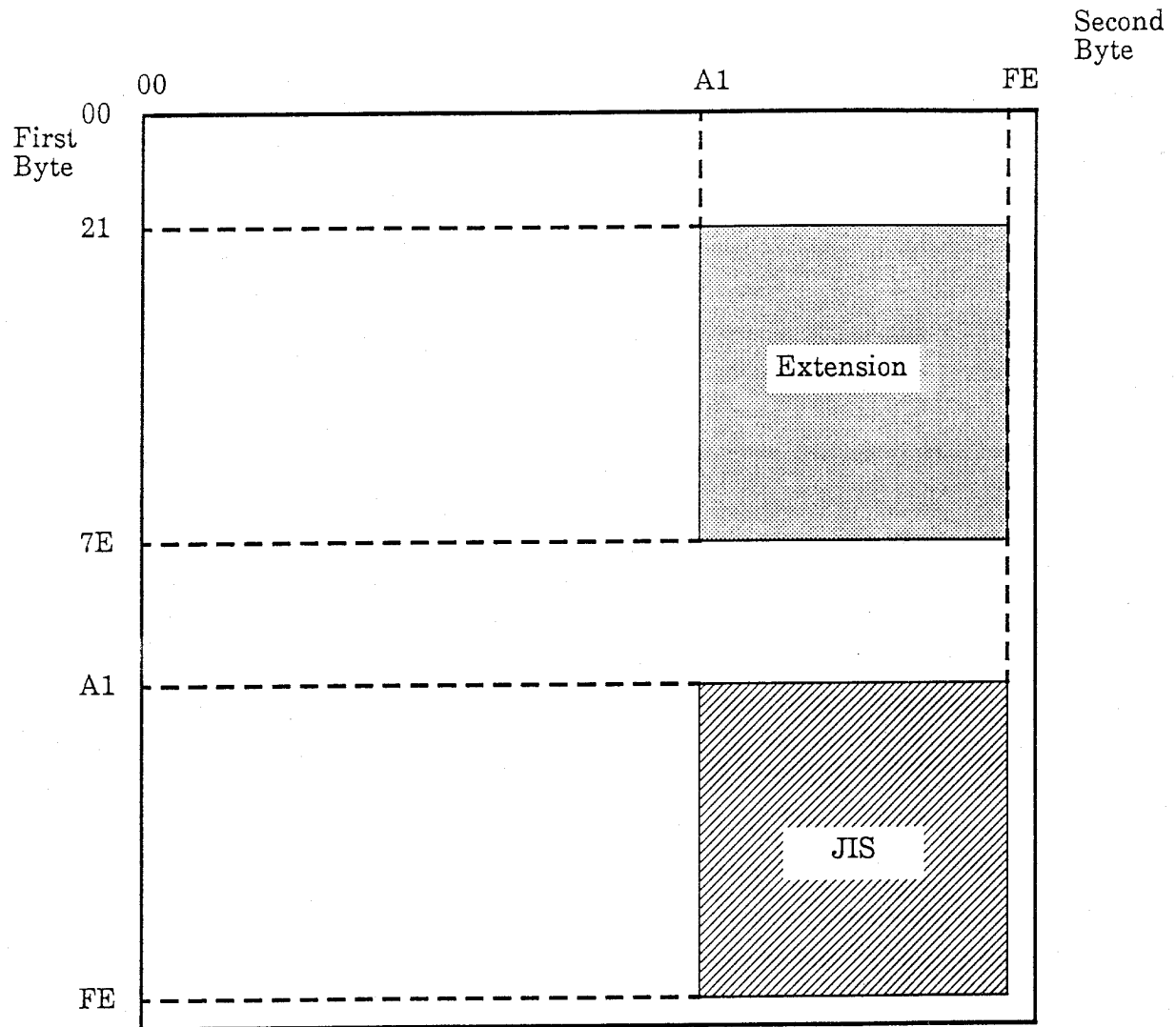
垂哀愛惡握匠汲安暗案以位依偉困委威尉意慰易為異移維緯衣違遺醫井域育一  
 老逸稱芋印圓姻引飲院陰韻右宇羽雨湍運雲營影映永泳英衛詠井液疫益  
 駢悅謁越閱園宴延援沿演煙猿綠遠鉛塩奧往應橫歐翁黃沖億屋  
 憶乙卸恩溫音下化何佳加可嫁家寡科暇央架歌河火禍箇華課  
 過蚊我画芽雅餓介會解回塊怪悔懷改岳海灰界皆繪開害慨  
 涯街該垣嘛各格核覓獲確寬角較郭悶學械樂額掛割渴滑簡  
 株刈乾冠寒刊勘勸卷喚堪官干眼幹患感願企危喜器欺汗管  
 棋棄機歸氣汽休協緊弓強近金徑兼限効厚肯航恨載算紫軸七  
 脚虐逆丘共凶筋協緊弓強近金徑兼限効厚肯航恨載算紫軸七  
 京虐逆丘共凶筋協緊弓強近金徑兼限効厚肯航恨載算紫軸七  
 軍供斤郡係深嚴幻護皇黑歲血旨治勺秀春唱訟食迅政切  
 元供斤郡係深嚴幻護皇黑歲血旨治勺秀春唱訟食迅政切  
 港原深嚴幻護皇黑歲血旨治勺秀春唱訟食迅政切  
 國語誤誤黑歲血旨治勺秀春唱訟食迅政切  
 才溝穀探殺支次邪愁述召肖色甚性赤選操速隊壇注賃敵  
 札探殺支次邪愁述召肖色甚性赤選操速隊壇注賃敵  
 指殺支次邪愁述召肖色甚性赤選操速隊壇注賃敵  
 時殺支次邪愁述召肖色甚性赤選操速隊壇注賃敵  
 蛇殺支次邪愁述召肖色甚性赤選操速隊壇注賃敵  
 術殺支次邪愁述召肖色甚性赤選操速隊壇注賃敵  
 升殺支次邪愁述召肖色甚性赤選操速隊壇注賃敵  
 紹殺支次邪愁述召肖色甚性赤選操速隊壇注賃敵  
 職殺支次邪愁述召肖色甚性赤選操速隊壇注賃敵  
 尋殺支次邪愁述召肖色甚性赤選操速隊壇注賃敵  
 征殺支次邪愁述召肖色甚性赤選操速隊壇注賃敵  
 踐殺支次邪愁述召肖色甚性赤選操速隊壇注賃敵  
 插殺支次邪愁述召肖色甚性赤選操速隊壇注賃敵  
 足殺支次邪愁述召肖色甚性赤選操速隊壇注賃敵  
 逮殺支次邪愁述召肖色甚性赤選操速隊壇注賃敵  
 注殺支次邪愁述召肖色甚性赤選操速隊壇注賃敵  
 賃殺支次邪愁述召肖色甚性赤選操速隊壇注賃敵  
 敵殺支次邪愁述召肖色甚性赤選操速隊壇注賃敵  
 動殺支次邪愁述召肖色甚性赤選操速隊壇注賃敵  
 日殺支次邪愁述召肖色甚性赤選操速隊壇注賃敵  
 背殺支次邪愁述召肖色甚性赤選操速隊壇注賃敵  
 伴殺支次邪愁述召肖色甚性赤選操速隊壇注賃敵  
 碑殺支次邪愁述召肖色甚性赤選操速隊壇注賃敵  
 敏殺支次邪愁述召肖色甚性赤選操速隊壇注賃敵  
 腹殺支次邪愁述召肖色甚性赤選操速隊壇注賃敵  
 編殺支次邪愁述召肖色甚性赤選操速隊壇注賃敵  
 芳殺支次邪愁述召肖色甚性赤選操速隊壇注賃敵  
 奔殺支次邪愁述召肖色甚性赤選操速隊壇注賃敵  
 予殺支次邪愁述召肖色甚性赤選操速隊壇注賃敵  
 役殺支次邪愁述召肖色甚性赤選操速隊壇注賃敵  
 庸殺支次邪愁述召肖色甚性赤選操速隊壇注賃敵  
 利殺支次邪愁述召肖色甚性赤選操速隊壇注賃敵  
 力殺支次邪愁述召肖色甚性赤選操速隊壇注賃敵  
 路殺支次邪愁述召肖色甚性赤選操速隊壇注賃敵

Figure 4.11: The Joyo Kanji character table

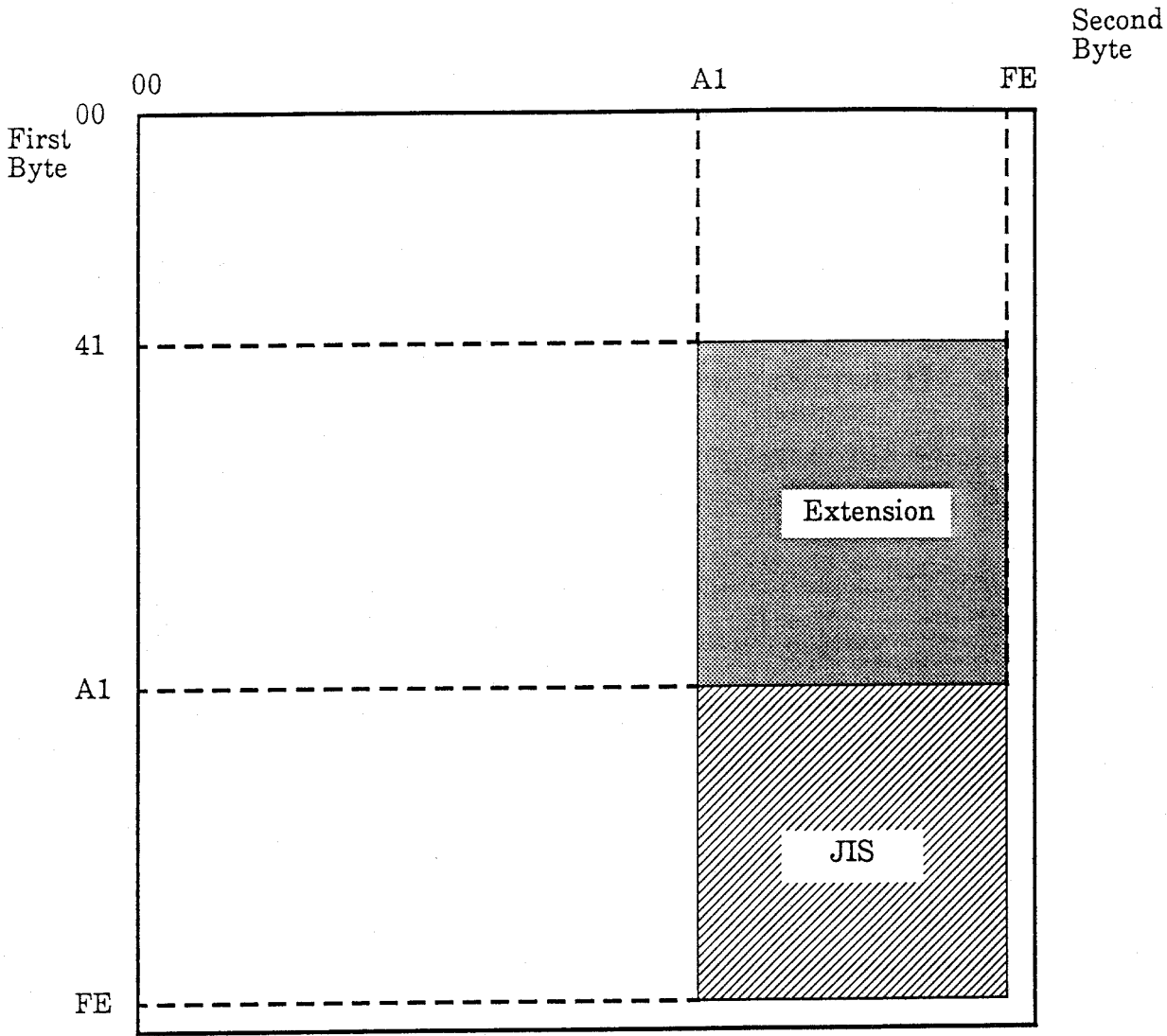


IBM EBCDIC DBCS CODE TABLE

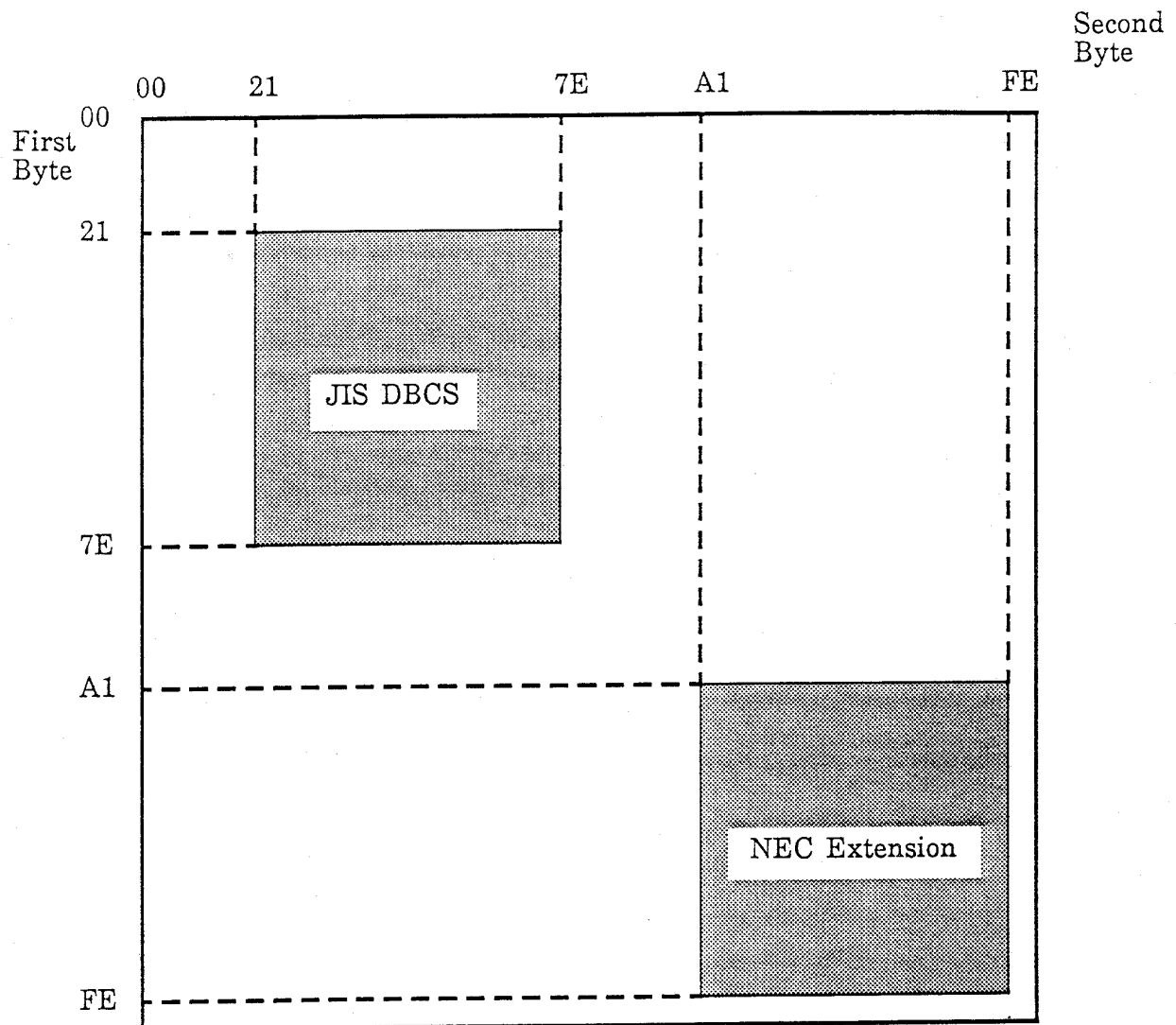




LETS-J JIS DBCS Code Table

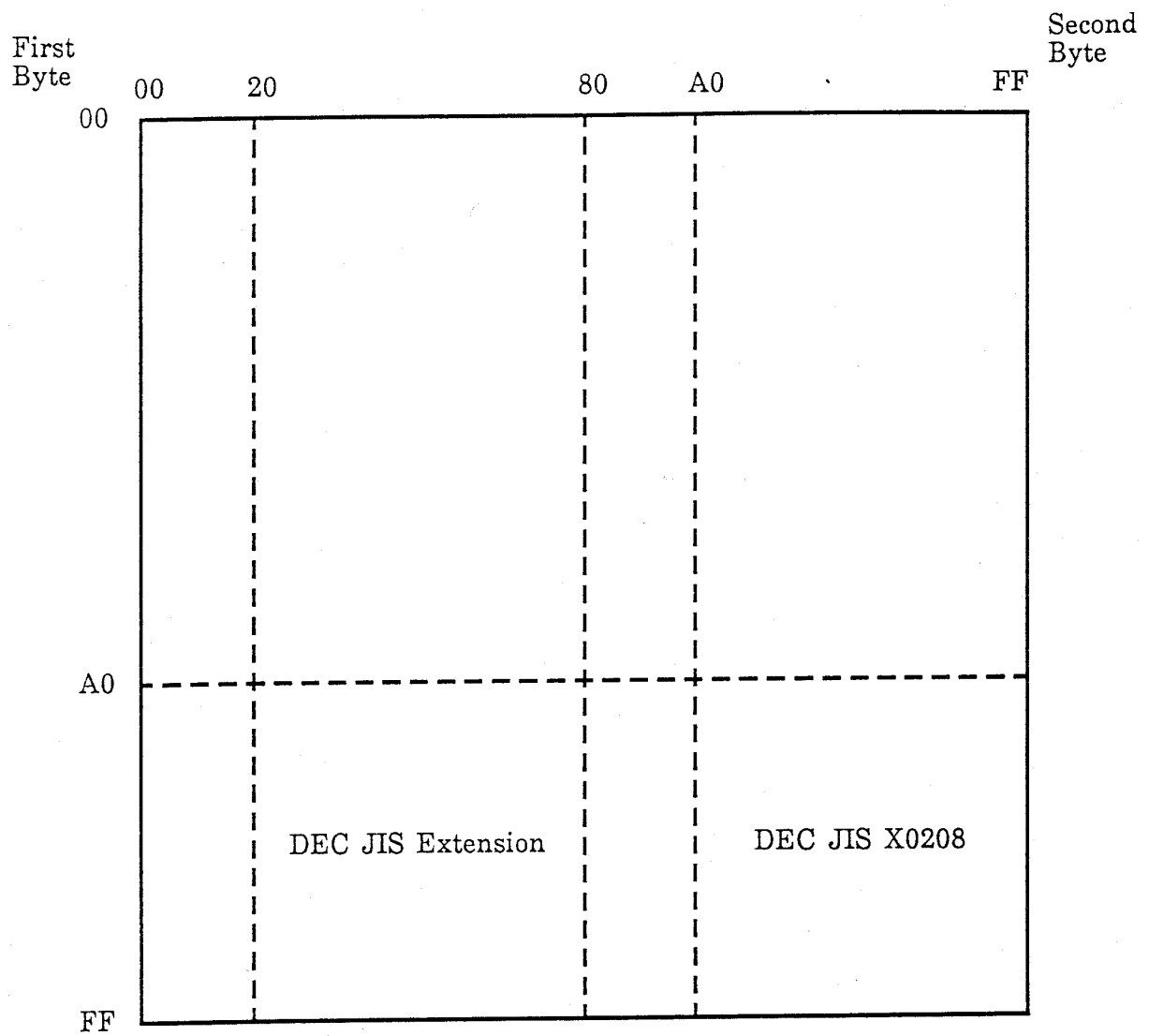


JBIS, Fujitsu & Hitachi JIS DBCS Code Table



NEC JIS DBCS Code Table





DEC DBCS Code Table

First Byte	Second Byte	
	00	40 7E 80 FC
00		
81		
9F		
E0		
FC		

Shift JIS DBCS CODE TABLE

Figure 4.18:

#### 4.2.4 DBCS Problems

If a system uses SBCS and DBCS character sets you have to be aware of some arising problems. If you use only SBCS characters you could work like in the "normal" ASCII environment (performing byte oriented operations). It is the same if you work only with DBCS characters. The trouble begins if you start to use mixed strings containing SBCS and DBCS characters.

If your system (operating system, programming language, application) is able to handle both types of characters you have to be aware that the Japanese DBCS characters (Zenkaku) have the double width of a standard ASCII SBCS or JIS SBCS (Hankaku). If you, e.g., design screen masks or reports you have to consider that the Japanese user could enter both SBCS and DBCS. *In the following example's S refers to a SBCS character and DD refers to a DBCS character.*

First an example of a SBCS character string :

S	S	S	S	S
---	---	---	---	---

(length 5 Bytes)

Now a five characters long DBCS string :

DD	DD	DD	DD	DD
----	----	----	----	----

(length 10 Bytes)

Here you will see a five character long mixed string :

S	S	DD	S	DD
---	---	----	---	----

(length 7 Bytes)

As mentioned above the length of all strings is five characters, but the actual length of the displayed string is different. If you design a screen mask or a report you have to take this in consideration and make your design so that it is able to handle all three types of strings (SBCS, DBCS and mixed strings). A programming language or application has to be able to handle strings with SBCS and DBCS. For example, the database system Oracle, uses no special data type for DBCS characters. The Japanese version of Oracle is able to handle DBCS characters. If you create a table with a 20 character long data field, this field could accept SBCS and DBCS characters as input. You could enter up to 20 SBCS characters or up to 10 DBCS characters (2 byte for each character). If you enter a mixed SBCS/DBCS string the maximum length can not exceed the length of 20 byte (a mixed string could contain, e.g., 10 SBCS and 5 DBCS characters which represent a 20 byte long string). This strategy has the

advantage that you not have to change the definition of a table. In addition the screen mask and report layout have not to be redesigned. It gets even more difficult if your system uses SI/SO (KI/KO) sequences. The same example with SI/SO sequence would look like this :

The first example with a five character SBCS string :

S	S	S	S	S
---	---	---	---	---

 (length 5 Bytes)

Now again a five characters long DBCS string with SI/SO sequence :

$\begin{smallmatrix} S \\ I \end{smallmatrix}$	DD	DD	DD	DD	DD	$\begin{smallmatrix} S \\ O \end{smallmatrix}$
------------------------------------------------	----	----	----	----	----	------------------------------------------------

 (length 12 Bytes)

A five character long mixed string with SI/SO sequence would look like :

S	S	$\begin{smallmatrix} S \\ I \end{smallmatrix}$	DD	$\begin{smallmatrix} S \\ O \end{smallmatrix}$	S	$\begin{smallmatrix} S \\ I \end{smallmatrix}$	DD	$\begin{smallmatrix} S \\ O \end{smallmatrix}$
---	---	------------------------------------------------	----	------------------------------------------------	---	------------------------------------------------	----	------------------------------------------------

 (length 11 Bytes)

You will recognize that the strings become longer through the use of the SI/SO sequence. The SI/SO sequence will not be displayed on the screen, but it makes the string, at least two bytes (depending on the implementation of the vendor), longer. In addition the use of a SI/SO sequence makes it more difficult to handle, e.g., string operations. These operations are operations like cursor movement, backspace, wrapping of strings, deletion or insertion and windowing.

First I will explain the handling of strings with a SI/SO sequence and later I will give you some examples about the handling of mixed strings without SI/SO sequence. The difference between the handling is that you have to be aware of the SI/SO sequence. It is easier to handle strings without SI/SO sequence but also here you have to be aware of the difficulties in the handling of mixed strings.

### Handling of mixed strings with SI/SO sequence

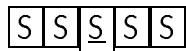
If you press a cursor key to move forward or backward through a string the system has to increase or decrease the pointer of the actual cursor position. This would look like this :

S	S	<u>S</u>	S	S
---	---	----------	---	---

cursor right ( $\rightarrow$ ) pressed once

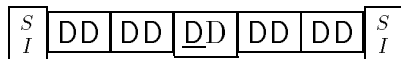
S	S	S	<u>S</u>	S
---	---	---	----------	---

cursor left ( $\leftarrow$ ) pressed once

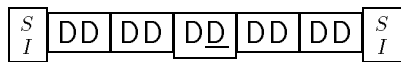


This works quite fine in a SBCS environment which works byte oriented. If we now have a look at the DBCS string, we will see that we no longer can use the byte oriented approach. In order to handle DBCS correctly we have to use a character oriented approach :

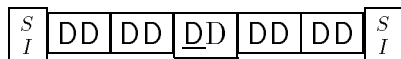
byte processing



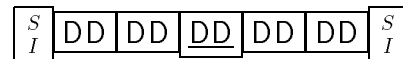
cursor right ( $\rightarrow$ ) pressed once



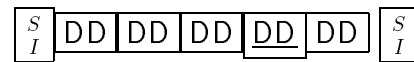
cursor left ( $\leftarrow$ ) pressed once



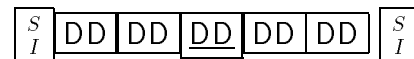
character processing



cursor right ( $\rightarrow$ ) pressed once



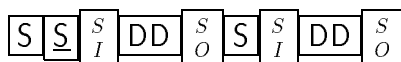
cursor left ( $\leftarrow$ ) pressed once



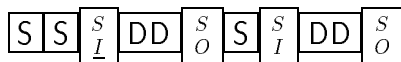
In the byte oriented approach the cursor does not move to the next character if you press it once. It moves from the first byte of the DBCS character to the second byte. Only in the character oriented version the cursor moves to the next character.

This gets even more complicated if you have to handle mixed strings which contain SBCS and DBCS characters :

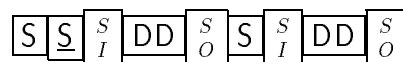
byte processing



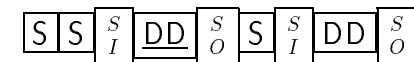
cursor right ( $\rightarrow$ ) pressed once



character processing



cursor right ( $\rightarrow$ ) pressed once

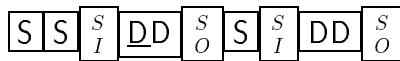


In the byte oriented example the cursor moves to the next byte, the SI sequence, and not to the next character. In the character oriented system the routine which moves the cursor has to detect the start of a DBCS character by recognizing the SI sequence. Then the routine has to set the cursor to the first byte of the DBCS character.

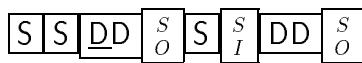
Similar problems occur if we press the backspace key to delete a character. In the next example you will see that in the byte oriented version the SI sequence gets lost and

leaves us with a garbage string. This string will no longer displayed correct because the SI sequence is missing and so the system is not able to recognize the start of the DBCS characters. Depending on the implementation the system will try to display the bytes of the DBCS character as SBCS characters (e.g., as Katakana or ASCII character depending on the code).

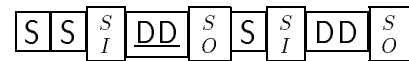
byte processing



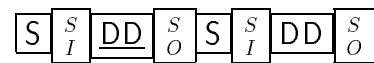
Backspace ( $\Leftarrow$ ) pressed once



character processing



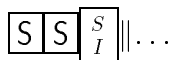
Backspace ( $\Leftarrow$ ) pressed once



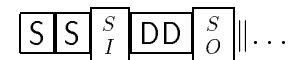
The same type of problem will occur if you truncate (e.g., the third character), exchange a character or insert a character (or string) in the mixed string.

Truncation after the third character

byte processing

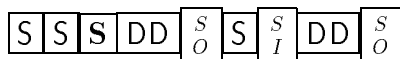


character processing

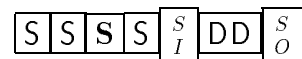


Exchange of a SBCS character at the third position

byte processing



character processing

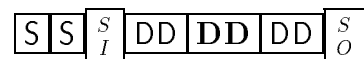


Exchange of DBCS character at the fourth position

byte processing

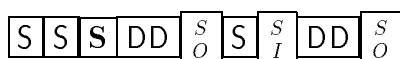


character processing

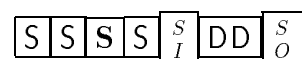


Insertion of a SBCS character at the third position

byte processing



character processing



Insertion of a DBCS character at the second position



As you may recognize the routines have to take care not only of the DBCS characters but also of the SI/SO sequence.

This breed of problem will also affect wrapping a string, searching a character in a string, upper and lower casing or working with windows on a screen. In the case of an environment which works with SI/SO sequences the handling of mixed strings is more complicated than in a system which works without SI/SO sequences. Nevertheless it is necessary to be aware of the mixed string problems in a, e.g., Shift-JIS environment.

### Handling of mixed strings without SI/SO sequences

Again I will start with the cursor movement. In picture A in figure 4.19 on page 93 (all examples [24]) you will see the cursor movement in a byte oriented environment. All characters in the picture are DBCS characters. There is no SI/SO sequence required because the MSB of this type of DBCS characters is always set to one. If we now press to cursor right key the cursor will move one byte to the right. From the first byte of a DBCS character (in line 1) to the second byte of the DBCS character (in line 2) and so on. In picture B (in figure 4.19) you will see how the cursor will move if the system works character oriented instead of byte oriented. The cursor jumps from one character to the next DBCS character. If we had a mixed string the system could distinguish between SBCS and DBCS characters by looking at the MSB. If the MSB is zero the pointer for the cursor position has to be moved one byte. If the MSB is one the system has to process a DBCS character and has to move the pointer for the cursor position two bytes instead of one byte.

As mentioned above you have to check which type of character you will process, e.g., if you press the backspace key. If have a DBCS character string and the system works in byte oriented mode the press of the backspace key will delete just one byte of the string. This will leave us with a corrupted string (see picture A in figure 4.20 on page 94). As you see some of the characters will change because the byte pairs where change through the byte oriented backspace operation. On the opposite this will not happen if you use a character oriented operation. In this case the system would correctly delete one DBCS character, i.e., one byte pair.

On page 95 in figure 4.21 you will see in picture A the byte oriented replace operation. After typing the (SBCS) character " H " the system will replace just one byte and so the mixed string will be corrupted. The Kanji character CBDC<sub>H<sub>ex</sub></sub> (Hon or Moto) will partly be replaced by the " H " (48<sub>H<sub>ex</sub></sub>). The character " D " (44<sub>H<sub>ex</sub></sub>) will form with the second byte from the DBCS character a new (illegal) DBCS character. If the system works in the character oriented mode the replace operation will replace the Hon (or Moto) Kanji character with either one byte and deletes the second byte of the DBCS character or it replaces the first byte with the code for " H " (48<sub>H<sub>ex</sub></sub>) and the second byte of the DBCS with the SBCS code for space (20<sub>H<sub>ex</sub></sub>).

The handling of mixed character string's effects all string handling operations of a computer system. For example if we use the operations for upper or lower casing the system has to distinguish between byte processing and character processing. In picture A in figure 4.22 (page 96) you will see the upper casing (No. 1) and lower casing (No. 2) operation performed in the byte oriented mode. This will corrupt the string. If we use the character oriented operations the string will be processed correctly (see picture B, No.1 & 2 in figure 4.22). A group of similar problems are the string wrapping and substring operations. On page 97 in figure 4.23 you will see what happens when the system has to perform a line-wrap (e.g., at the end of a line). If the space which is left at the line is 18 bytes and the string is longer then this space (here 21 bytes) the system has to wrap the line and put some of the characters in the next line. Is this operation performed byte oriented the system would wrap incorrectly and would split a DBCS character (picture A in figure 4.23). This would cause that the first character in the next line is incorrect. If the system works character oriented it would use only 17 of the 18 bytes and would split the string correct (picture B in figure 4.23). The same problem appears if we try to move a longer string into a shorter string data field. In the example in figure 4.24 on page 98 the original string is 21 Bytes long. If a byte oriented routine moves this string into a 10 byte long data field the system would cut the string after 10 bytes and would leave us with a corrupted DBCS character (picture A). The character oriented operation instead would copy only 9 byte because the next character after the C is a DBCS character which would not fit in the remaining space.

As last example for the splitting of DBCS characters I will show you a windowing operation performed in byte oriented and character oriented mode. In picture A on



page 99 (figure 4.25) you will see that the DBCS character is " split ". In a GUI this would not cause any problems but if you work with a character oriented program (in text mode) you have to be careful where you place the border of your window. You can not split SBCS character (because they are only one byte long and placing a border above it will always hit the whole character) but if you place the border of a window on the second byte of a DBCS character this character will be corrupted. To perform the windowing operation correctly in a DBCS environment you (or your program) have to watch where it places the border of a window (see picture B in figure 4.25).

Another function which could give us funny results is the search string function. If you perform a byte oriented search operation on the source string in figure 4.26 (page 100) with the parameter `C7A5Hex` the function will return 2 positions in the source string (picture A). One of this positions is the second byte of the third DBCS character and the first byte of the fourth DBCS character. The correct result is that only the last DBCS character matches the search string (picture B).

If you use DBCS characters with or without SI/SO sequence you always have to be aware of the arising problems in handling these characters. If you use the byte oriented operations of an English or American system you will get some strange results. Only if you use the correct processing method (character oriented) your system will give you the results which you expect.

日本ディジタルイクイップメント株  
日本ディジタルイクイップメント株  
日本ディジタルイクイップメント株  
日本ディジタルイクイップメント株

A) Byte processing

日本ディジタルイクイップメント株  
日本ディジタルイクイップメント株  
日本ディジタルイクイップメント株  
日本ディジタルイクイップメント株

B) Character processing

Figure 4.19: Cursor Movement

日本 デ ィ ジ タ ル 株 式 会 社

C6FC CBDC A5C7 A5A3 A5B8 A5BF A5EB B3F4 BCB0 B2F1 BCD2

日本 デ ィ ジ タ コ  芦 饅

C6FC CBDC A5C7 A5A3 A5B8 A5BF A5B3 F4BC B0B2 F1BC D2

A) Byte processing

日本 デ ィ ジ タ ル 株 式 会 社


C6FC CBDC A5C7 A5A3 A5B8 A5BF A5EB B3F4 BCB0 B2F1 BCD2

日 本 デ ィ ジ タ 株 式 会 社

C6FC CBDC A5C7 A5A3 A5B8 A5BF B3F4 BCB0 B2F1 BCD2

B) Character processing

Figure 4.20: Backspace operation

日	本	D	E	C	株	式	会	社
C6FC	CBDC	44	45	43	B3F4	BCB0	B2F1	BCD2
日	H		E	C	株	式	会	社
C6FC	48	DC44	45	43	B3F4	BCB0	B2F1	BCD2

## A) Byte processing

日	本	D	E	C	株	式	会	社
C6FC	CBDC	44	45	43	B3F4	BCB0	B2F1	BCD2
日	H	D	E	C	株	式	会	社
C6FC	48 (20)	44	45	43	B3F4	BCB0	B2F1	BCD2

## B) Character processing

Figure 4.21: Replace function

日 本 デ ッ ク 株 式 会 社 d E c

C6FC CBDC A5C7 A5C3 A5AF B3F4 BCB0 B2F1 BCD2 64 45 63

Text before processing

1) 頓 本 デ ッ ク 郭 式 英 社 DEC

C6DC CBDC A5C7 A5C3 A5AF B3D4 BCB0 B2D1 BCD2 44 45 43

2) 脣 潜 ヨ ヤ ク 株 式 会 酒 d e c

E6FC EBFC A5E7 A5E3 A5AF B3F4 BCB0 B2F1 BCF2 44 65 63

A) Byte processing

1) 日 本 デ ッ ク 株 式 会 社 D E C

C6FC CBDC A5C7 A5C3 A5AF B3F4 BCB0 B2F1 BCD2 44 45 43

2) 日 本 デ ッ ク 株 式 会 社 d e c

C6FC CBDC A5C7 A5C3 A5AF B3F4 BCB0 B2F1 BCD2 64 65 63

B) Character processing

Figure 4.22: Upper and Lower casing

日 本 デ D E C ッ ク 株 式 会 社

C6FC CBDC A5C7 44 45 43 A5C3 A5AF B3F4 BCB0 B2F1 BCD2

Text before processing

日 本 デ D E C

C6FC CBDC A5C7 44 45 43

A5

A) Byte processing

日 本 デ D E C

C6FC CBDC A5C7 44 45 43

B) Character processing

Figure 4.23: Word wrap

日 本 デ ッ ク D E C 株 式 会 社  
 C6FC CDBC A5C7 A5C3 A5AF 44 45 43 B3F4 BCB0 B2F1 BCD2 (21 BYTE)

Text before processing

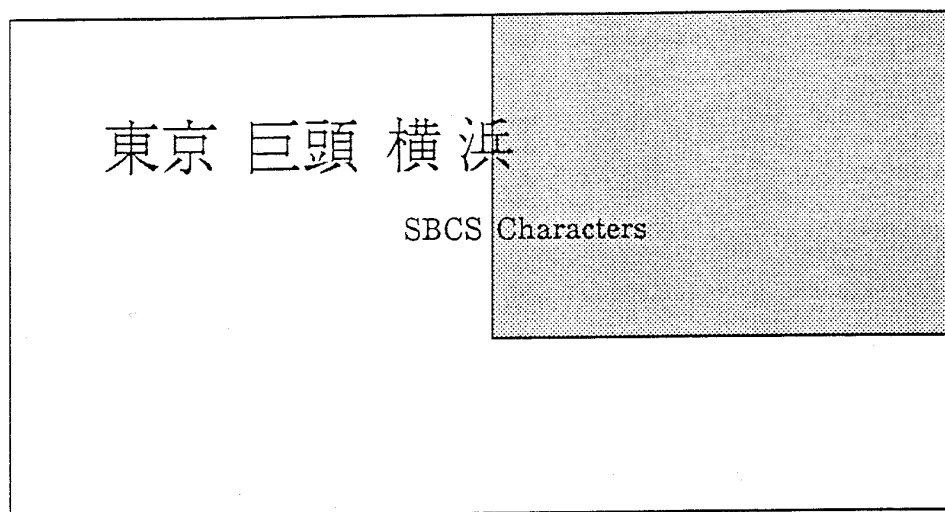
日 本 デ ッ ク D E C 株 式  
 C6FC CDBC A5C7 A5C3 A5AF 44 45 43 B3F4 BCB0 B2 (18 Byte)  
 僅  
 F1BC D2

A) Byte processing

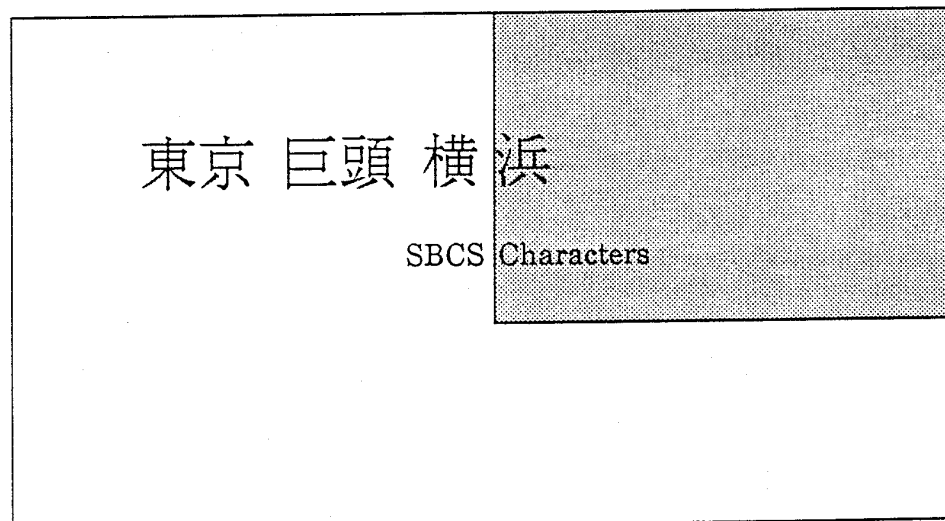
日 本 デ ッ ク D E C 株 式  
 C6FC CDBC A5C7 A5C3 A5AF 44 45 43 B3F4 BCB0 (17 BYTE)  
 会 社  
 B2F1 BCD2

B) Character processing

Figure 4.24: Line truncation



A) DBCS Character split by Window



B) Correct DBCS Character windowing

Figure 4.25: DBCS windowing



日本デック株妊  
C6FC CBDC A5C7 A5C3 A5AF B3F4 C7A5

Source String

Search String : 妊 (C7A5)

日本デック株妊  
C6FC CBDC A5C7 A5C3 A5AF B3F4 C7A5

A) Byte processing

日本デック株妊  
C6FC CBDC A5C7 A5C3 A5AF B3F4 C7A5

B) Character processing

Figure 4.26: Search string function

## 4.3 Number & Currency convention

In the Japanese business world many computer programs rely on numbers, monetary values and the proper handling of this kind of information are very important.

### 4.3.1 Number representation

The presentation of numbers varies only in the use of different separators for a group of three digits. Table 4.4 (on page 101, [1]) shows you an example of different formats for the representation of numbers. As you see in the table there are only

Country	Positive Numbers	Negative Numbers
Germany	12.345,67	-12.345,67
Australia	12,345.67	-12,345.67
Arabic	123.45,67	123.45,67-
France	12 345,67	-12 345,67
South Africa	12 345.67	(12 345.67)
<b>Japan</b>	<b>12,345.67</b>	<b>-12,345.67</b>

Table 4.4: Number Notations

few variations in the representation of a number. As separators are used the period, comma and space. The minus is mainly put in front of a negative number.

In addition to the western way of writing numbers the Japanese use their traditional Kansuji characters (see figure 4.27 on page 103). In this way of writing the Japanese use special Kanji characters to express 10 (Ju), 100 (Hyaku), 1,000 (Sen, do not mix up with Sen, each Sen has a different Kanji character), 10,000 (Man), 100,000,000 (Oku) and 1,000,000,000 (Cho).

For expressing fractions, ordinal numbers and percentage the Japanese use the western way and their traditional way. A fraction  $\frac{N}{M}$  is written in Japanese with the Bun No as the fraction stroke (see picture b in figure 4.28 on page 109). Ordinal numbers are written with the Bam-me as sign (see picture c in figure 4.28). For expressing a percentage the Japanese use Wari ( $\frac{1}{10}$ ), Bu ( $\frac{1}{100}$ ) and Rin ( $\frac{1}{1000}$ ). If you want to

express 56.7 % you could write it like in picture d in figure 4.28. All of these numbers will be written with western style numbers or the traditional Kansuji characters (see figure 4.27 on page 103).

### 4.3.2 Rounding of Numbers

For business purposes the Japanese use the normal way of rounding numbers, i.e., from XXX.0 to XXX.4 the system must round to XXX. In the case of that the number is between XXX.5 and XXX.9 the system has to round to XX(X+1). For example : 123.454 rounds to 123.45 and 123.457 rounds to 123.46 ([1]). In some business areas it is common to round to the next 1,000 or 10,000 Yen ([11]).

### 4.3.3 Currency

The Japanese currency is called Yen (or En). One Yen is 100 Sen. Today the Sen is mainly used in the banking area. Yen is represented as a one byte character (SBCS) and En is a two byte (DBCS) character. In everyday life the Japanese use only the Yen. The effect is that there are no currency decimal positions used in Japan (One Yen is the smallest coin and 10,000 Yen the biggest bill). The international representation for the Japanese Yen is, regarding to ISO 4217 (Codes for the representation of currency and funds), the JPY (see table D in figure 4.32 on page 114) symbol. The small difference in the use of Yen and En is that the Yen is placed in front of the amount and the En is placed behind the amount of money. For an example for the writing of positive and negative amounts of money see table D in figure 4.32. The representation of positive and negative values follows the standard for writing numbers mentioned above.

For a data field which represent a monetary value you should reserve a currency field length among 12 and 15 digits ([1], [11]). In this field you have to put the Yen (SBCS) or En (DBCS) symbol and a comma every three digit.

Number	Name	Name in Kanji	Name in Kansuji
0	Zero, Rei	零	〇
1	Ichi	壹	一
2	Ni	貳	二
3	San	参	三
4	Yon, Shi	四	四
5	Go	五	五
6	Roku	六	六
7	Nana, Shichi	七	七
8	Hachi	八	八
9	Kyu	九	九
10	Ju	十	十
100	Hyaku	百	百
1,000	Sen	千	千
10,000	Man	万	万
100,000,000	Oku	億	億
1,000,000,000,000	Cho	兆	兆

Table of Kanji and Kansuji Characters of Japanese Numbers

For example, 1992 is written 一千九百九十二

Figure 4.27:

## 4.4 Date & Time convention

In this section I would like to introduce the reader to the different date and time formats which are used in Japan. Furthermore I will explain the different calendar systems that are used in Japan.

### 4.4.1 Date formats

In this section I would like to introduce the reader to the differences in the representation of a date. In some cultures (e.g., Taiwan, Thailand, Arab countries, Israel, Japan) is not only the Gregorian calendar system used. Besides this calendar these countries use one or more other calendar systems based on their culture or religion (see, e.g., [1], 5-2).

In Japan there are two types of calendar systems commonly used :

- The Gregorian calendar system The first system is our commonly used Gregorian calendar system. With 365 days a year, 12 months with 30, 31 or 28 (in a leap year 29) days. The difference between the date convention in, e.g., Germany, is that the Japanese got a different style of writing a date, here you will see some examples<sup>2</sup> for different styles of date representation (see table 4.5).

Country	Representation	Example
Germany	DD . MM . YYYY	10.02.1966
Australia	DD / MM / YYYY	26/07/1991
USA	MM - DD - YYYY	03-20-1992
<b>Japan</b>	<b>YYYY - MM - DD</b>	<b>1992-03-26</b>

Table 4.5: Gergorian Date represantation

The system, which is used in Japan is the year - month - day representation. This way got the advantage that, if you want to bring something in chronologic (or reverse chronologic) order, it is very easy to do.

---

<sup>2</sup>DD refers to the day, MM to the month and YYYY refers to the year

- Gengou calendar system (see picture a in figure 4.29 on page 111)

The second heavily used system (e.g., from the Japanese government, the most governmental forms expect this date format) is based on the reign of the Japanese tenno (emperor). After the death of the emperor a commission decides the name for the next era. Then this new era starts from the year one. Shouwa was the name of the last era, which ended 1989, in the 64<sup>th</sup> year of this era, with the death of emperor Hirohito. So that today, in the year 1992, is the 4<sup>th</sup> year of the actual tennos reign. His name is Akihito and the actual era is called the Heisei era. (*Historic note : in this date format the Japanese use the terms Seireki and Kigenzen for dates before the Meiji era, see figure 4.28 on page 109*). This Gengou (sometimes also called Nengou) type of calendar system gives the computer systems and software some problems on the way :

- The system should ” know ” if there is a coronation of a new emperor, so that the name of the era could be changed. Also the counter for the year has to be set to one. An operator has to enter this information when a change of the emperor occurs. Also the system should provide a function to transfer from the Gregorian calendar system to the Gengou calendar system and reverse.
- The printed or displayed presentation of the Gengou calendar system is possible in several ways :
  - \* A short, modern form is EY - MM - DD, like 4 - 4 - 17, which is e.g., the 4<sup>th</sup> year of the era, April, the 17<sup>th</sup>. EY represents the year of emperors reign, the era year. This form is the easy everyday used form for, e.g., tickets, advertising. Besides that form there is a slightly different form used. (see picture b-1 in figure 4.29 on page 111) In front of the EY (emperor’s year) is printed (or displayed) the first letter of the era name, like H 4 - 4 - 17. So that it is easier to see which era is meant. In the first example it does not have to be a date in the Heisei era, it also could be a date in, e.g., the Meiji era.
  - \* An other, often used form, is the EY KY MM KM DD KD form as date representation. With the era year and the Kanji Nen (KY, year), then the month and the Kanji Tsuki (KM, month) and at last the day with the Kanji Hi (KD, day). This date representation would look for

the 16<sup>th</sup> of April, 4<sup>th</sup> year of the era, like in picture b-2 in figure 4.29 on page 111.

Sometimes, e.g., cash registers use a similar form. Instead of the EY (emperor year) you will see the Gregorian calendar system year. This mixed form is sometimes used, because you do not have to change the sign for the era. It still uses the Kanji characters for year, month and day.

- \* The third form is an extended version of the second form. For this form the system has to add the name of the era in Kanji characters in front of the second form. In the year 4 of the Heisei era (1992) , at April the 16<sup>th</sup>, this form would look like in picture c in figure 4.29 on page 111.
- \* Besides this two advanced ways it is possible to use the Kansuji for the month (see figure 4.27 on page 103 or the Kanji characters for the Japanese name of the month. You will find a table with this representation of the months in figure 4.30 on page 112). Today this form of date representation is a little bit unfashionable, but it could be used under some circumstances (see picture d in figure 4.29 on page 111). Nevertheless it is possible to write the whole date in Japanese Kanji characters (see picture e in figure 4.29 on page 111) in this case you could use the kansuji (see figure 4.27 on page 103) or the traditional Japanese name of the day (see figure 4.31 on page 113).
- Except for the first forms your system has to be able to display or print Kanji characters, if you want to use the advanced forms of date representation in the Gengou calendar system.

Besides the era year and the additional Kanji characters does this calendar version work like the Gregorian calendar system (12 months, a 30, 31 or 28 (29 in a leap year) days, see [1], 5-4). . In table 4.6 I will give a short overview about the last emperor eras in Japan.

The Kanjis for these eras would look like picture a in figure 4.28 on page 109.

Following the JIS standard JIS X0301-1977 (replaced this year with JIS X0301-1992) the JSA recommends three different time formats (see [29]) :

1. For EDI purposes the format YYMMDD (920326) or YYYYMMDD (19920326).
2. For the human machine interaction the JSA recommends the form YY-MM-DD (92-03-26) or YYYY-MM-DD (1992-03-26). A second recommended specification is the YY MM DD or YYYY MM DD form.
3. The form, which corresponds to the Japanese calendar, is YY.MM.DD (4.03.26) or *era*YY.MM.DD (H4.03.26, with H = Heisei, S = Showa, T = Taisho and M = Meiji)

#### 4.4.2 Time formats

Like the most other countries in the World the Japanese use for business purposes the 24 hour time system (see [1], 5-5). Again, the difference is based on the representation of the data. But in this case the difference is just a minor difference. The separator, which separates hour, minute, and second is varying. Some countries like e.g., Argentina, Denmark, Italy, use the period as separator (" . "). In the most other countries use the colon (" : ") instead. In table 4.7 (page 110) the time that is represented is 22 hours (10 pm), 42 minutes, 00 seconds and an additional 30 milliseconds, if the milliseconds appear in the used time format. The separator for "fractions of a second" should, referring to the ISO (International Standardization Organization) standards ISO 3307 and ISO 1000, be the same, which is used as decimal separator in the number representation format, for the country (i.d. [1], 5-5).

The represented forms could be :

- one digit for tenths of a second
- two digits for hundreds of a second
- three digits for thousandths of a second

Besides this, in the business world used time format, there is an other Japanese time format. This time format contains, like the Gengou date representation, some special Kanji characters. The characters are called Ji, which is used for the hour, and Fun, which represents the minutes. So that the time 22:42:00 would look like picture a in figure 4.32 on page 114.



Moreover that time format the Japanese also know a 12 hour representation. This twelfth hour format runs actually from 0 to 11. In that case the Kanji Gogo is used as sign for PM (12:00 – 24:00) and the Kanji Gozen is the sign for AM (00:00 – 12:00). In this type of time format 22:42 would look like picture b in figure 4.32 on page 114. At 10:42 in the morning (AM) it would look like picture c in figure 4.32 on page 114. As in the Gengou date representation your system has to be able to display or print Kanji characters if you want to use this time format.

Following the JIS standard JIS X0302-1977 (combined later this year with the new JIS X0301-1992) the JSA recommends two different time formats (see [30]) :

1. For EDI purposes the format hhmmss (224200).
2. For the human machine interaction the JSA recommends the form hh:mm:ss (22:42:00).

In addition to the already introduced date and time formats there are combined date and time formats recommended by the JSA (see [30]) :

1. For EDI purposes the format YYMMDDhhmmss (920326224200) or YYYYMMDDhhmmss (19920326224200).
2. For the human machine interaction the JSA recommends the form YY-MM-DD-hh:mm:ss (92-03-26-22:42:00) or YYYY-MM-DD-hh:mm:ss. The second recommendation is the (YY)YY MM DD hh:mm:ss form.
3. The Japanese calendar based form looks like (*era*)YY.MM.DD,hh:mm:ss (H4.03.26,22:42:00).

As a last information about the time format in Japan I would like to mention that the Japanese do not use daylight saving time. There is no need to set the clock forward or backward in spring and fall.

a)	Era	Kanji	Emperor
	Meiji	明治	Mutsuhito
	Taisho	大正	Yoshihito
	Showa	昭和	Hirohito
	Heisei	平成	Akihito
	Years before Meiji		
	Seireki(A.D.)	西暦	Anno Domini
	Kigenzen(B.C.)	紀元前	Before Christ
b)	Fractions		
	$\frac{N}{M}$ or N/M	M bun no N	M分のN
c)	Ordinal Numbers		
	Number + Bam-me	番号 + 番目	
d)	Percentage		
	Unit	Name	Kanji
	1/10	Wari	割
	1/100	Bu	分
	1/1000	Rin	厘
	e.g. gowari	rokubu	nanarin = $\frac{567}{1000}$
	5割	6分	7厘

Figure 4.28: Emperor eras and Numbers

Era	From	To	Duration	Tenno
Meiji	1868	1912	45	Mutsuhito
Taisho	1912	1926	15	Yoshihito
Shouwa	1926	1989	64	Hirohito
Heisei	1989	...	...	Akihito

Table 4.6: Japanese eras

Argentina	22.42.00
Denmark	22.42.00,03
Italy	22.42.00,030
Belgium	22:42:00,030
Greece	22:42:00.030
<b>Japan</b>	<b>22:42:00</b>

Table 4.7: Time formats

a) Gengou(Nengou) 元号(年号) Calendar as Kanji

b-1) 92-4-16 ; 1992-4-16 ; 4-4-16 ; H4-4-16

b-2) 4 Nen 4 Gatsu 16 Nichi

4年4月16日

c) Heisei 4 Nen 4 Gatsu 16 Nichi

平成4年4月16日

d) Like c) with Month as Kanji

平成4年四月16日(1)

平成4年卯月16日(2)

e) Like d) with Year & Day + Month as Kanji

平成四年四月十六日 (1)

平成四年卯月十六日 (2)

Figure 4.29: Date representation

Month	Name	Name in Kanji	Traditional Name	Traditional Name in Kanji
January	Ichigatsu	一月	Mutsuki	睦月
February	Nigatsu	二月	Kisaragi	如月
March	Sangatsu	三月	Yayoi	弥生
April	Shigatsu	四月	Uzuki	卯月
May	Gogatsu	五月	Satsuki	皐月
June	Rokugatsu	六月	Minazuki	水無月
July	Shichigatsu	七月	Fumitsuki	文月
August	Hachigatsu	八月	Hazuki	葉月
September	Kugatsu	九月	Nagatsuki	長月
October	Jugatsu	十月	Kannazuki	神無月
November	Juichigatsu	十一月	Shimotsuki	霜月
December	Junigatsu	十二月	Shiwasu	師走

Table of Japanese Names for the Month

Day	Name	Name in Kanji	Abbreviation	Abbreviation in Kanji	Meaning
Monday	Getsuyobi	月曜日	Getsu	月	Moon
Tuesday	Kayobi	火曜日	Ka	火	Fire
Wednesday	Suiyobi	水曜日	Sui	水	Water
Thursday	Mokuyobi	木曜日	Moku	木	Wood
Friday	Kin'yobi	金曜日	Kin	金	Gold
Saturday	Doyobi	土曜日	Do	土	Earth
Sunday	Nichiyobi	日曜日	Nichi	日	Sun

The Names of the Day in Kanji

Figure 4.31:

- f)
- | Input<br>Romaji     | me | i | ji | Kanji |
|---------------------|----|---|----|-------|
| Display<br>Hiragana | め  | い | じ  | 明治    |

Figure 4.32: Time representation

## 4.5 Kana & Kanji Input

It is understandable that the Japanese can not build keyboards with thousands of keys to enter their Kana and Kanji characters (old typewriters where build in a similar way, see figure 4.34 on page 121). For this reason they use different methods to enter these characters. Usually the input is handled by a program know as FEP (Front End Processor). This program accepts the user input and handles the necessary conversions to the appropriate codes. In the following sections I will describe some of these methods and the historic development to the status quo today.

### 4.5.1 Front End Processor

Nowadays there are different ways to enter Kana and Kanji symbols. In this section I would like to introduce the reader to some possible ways.

Today there are two common used ways :

- Romaji  $\Rightarrow$  Hiragana  $\Rightarrow$  Kanji
- Hiragana  $\Rightarrow$  Kanji

Both ways are available in different implementation (e.g., IBM, Ricoh, NeXT, Hitachi, ...), but the basic principal behind the function is always the same. The main difference between the different methods is that some companies (or their FEP) use a separate window or line on the screen to display the Kana or Kanji before the user accepts the presented character. After the user has accepted the offered choice, the characters will be transferred to their place in the application and displayed at the last cursor position.

The other widely used method is to display the input direct on the screen (at the actual cursor position) in the application. So that the user will see his input appearing exactly at the place where he wants to enter the characters (inline conversion).

#### Kana to Kanji Conversion

The first way of conversion (via Romaji) is just a step before the Hiragana to Kanji conversion. For the most people this way is easier to use because they are more familiar



with the normal standard " QWERTY " computer keyboard then with the Hiragana computer keyboard (see figure 4.35 on page 122, [28]). Hiragana (and Katakana, both are called Kana characters) is a syllable alphabet, so that it is possible to enter the syllables in Romaji (our alphabet), i.e., the Latin alphabet (a table with Romaji and the corresponding Katakana and Hiragana syllable is starting at page 127). After typing the syllables in Romaji the FEP converts the input and displays the appropriate Kana character. To enter the name of the city " Tokyo " by using the first way, we have to type " toukyou " (which is the correct spelling for Tokyo in Japanese). If we type this on a " QWERTY " computer keyboard and the FEP is switched to the Romaji to Hiragana conversion mode, on the screen would appear the spelling for " Tokyo " in Hiragana (see figure 4.32, page 114, picture e ).

Depending on the FEP conversion routine we could now take over the Hiragana characters by pressing a certain key (e.g., space, enter) or we could press the key which would start the conversion to Kanji characters.

Tokyo (or *toukyou*) converted to Kanji causes no problems, because there is only one Kanji which represents this spelling. After pressing the conversion key, on the screen would appear the Kanji characters for " Tokyo " (see figure 4.32, page 114, picture e ).

In the other case, if there are different Kanji characters for a certain spelling than we have to choose the right one. There are different ways to do this :

- by pressing the Kanji conversion key again and again until the right Kanji will appear, or
- requesting a table with all, for this spelling, appropriated Kanji characters from the system and then choosing the right one

To make it easier for the user, the most hardware vendors have developed a special set of keys which are added to a normal keyboard. A user is now able to select the FEP mode by pressing one key or a combination of shift, control or alt(ernate) and a special key to switch between the FEP modes. To give you an example you will find in figure 4.37 pictures of a Japanese IBM notebook model 55 and the corresponding German IBM model. If you direct your attention to the space key you will recognize that the Japanese space key is much smaller then the space key on the German

keyboard. Instead of a big space key you will see three keys which are used to switch between the different modes of the FEP.

One of this keys is the Kanji character conversion key. The other keys are used to switch between Hiragana, Katakana, ASCII and other input methods (like input by JIS code).

Let me give you an example for the use of this, if you want to enter the name of the Meiji era you have to type " meiji ". On the screen would appear the spelling of " meiji " in Hiragana (see figure 4.32, page 114, picture f).

If you press now the Kanji conversion key the system would offer the most common (or often used) Kanji characters for this spelling (see figure 4.32, page 114, picture f). This could be (or is) the right Kanji character for that case, but in some cases you need an other Kanji which also represents the spelling, but has a different meaning. These Kanji characters would look like picture a) in figure 4.38 on page 125 (for the input " meiji "). Some systems are able to display a table of all appropriate Kanji characters and the user can select the right one by pointing with a mouse cursor or high-lightening the Kanji character under cursor control.

The other possible way of entering the syllables " to u kyo u " for " Tokyo " is to type them direct on the Hiragana keyboard (see figure 4.35, page 122). But today the most people prefer the " QWERTY " type computer keyboard.

### **Development of FEP**

When the Japanese started to develop the FEP method they used a simple architecture. After the user entered the spelling of a Kanji character the FEP is looking in a Kanji database for the Kanji which fits this spelling. After the user has selected and accepted the offered Kanji characters the code of this Kanji character is passed through to the application.

The first versions of FEPs where only able to convert one Kanji character at a time. In order to get the Kanji characters for " Tokyo " the user had to type " tou " and convert this to the Kanji character for east. After that he had to enter " kyou " and convert this to the Kanji character for capital city (see picture a, figure 4.36, page 123). This, Tan (single) Kanji conversion called, way of entering characters was slow and not very sophisticated. The next step was that the FEP system was able

to understand the spelling of compound Kanji characters (like ” Tokyo ” which is a combination of the two Kanji characters for east and capital city). The Jukugo (Kanji compound) conversion enabled the user to enter ” toukyou ” and to convert this to the appropriate Kanji characters (see picture b, figure 4.36, page 123). The next improvement of the FEP was the Renbunsetsu (phrase) conversion which allows the user to enter whole phrases (or sentences) and convert them at once. This way of converting Kanji and Kana characters is quit difficult because the Japanese use not only Kanji characters in a sentence. For example, Hiragana is used for grammatical constructs and Katakana or Romaji is used for foreign words. The Renbunsetsu FEP must be able to judge which Hiragana characters have to be converted and which not. Furthermore, if there are several Kanji characters for a spelling, the system has to give the user a choice of Kanji characters (see picture c, figure 4.36, page 123) depending on the context of the sentence. This is quite more complicated and needs an intelligent conversion algorithm, much more intelligent then the conversion algorithm for compound Kanji character conversion. Today it is possible for an application to overtake control in which mode the FEP is. The FEP could be switched to ASCII mode if the application (or the programmer) wants that the user could enter only ASCII characters (e.g., for filenames). Likewise it can allow to user to choose which type of characters he wants to enter.

### Structure of FEPs

The basic structure behind all FEPs is similar. Some are a part of the keyboard driver, some work as a daemon in the background of a system. But all work in a similar way (for the basic structure see figure 4.33, page 118).

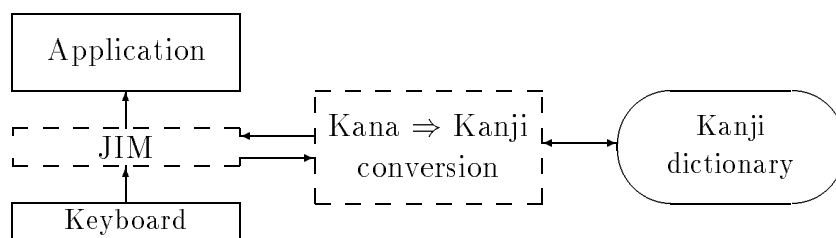


Figure 4.33: Japanese Input Manager

In order to do their job, converting a user input to Kanji character(s), the FEP needs a Kanji dictionary. In this dictionary is the pronunciation (Yomi in Japanese) for the Kanji characters listed. Depending on the mode the **J**apanese **I**nterface **M**anager (JIM) passes the user input through to the application or converts the input to Hiragana, Katagana or (via the Kana characters) to Kanji characters. Let us now have a closer look on the conversion from Romaji or Hiragana to Kanji. The normal way is that the system, after you entered the Romaji or Hiragana pronunciation, has a look into the Kanji character database to find an appropriate Kanji (or a link to the right code number of a Kanji or the Kanji characters which fit the spelling). After that the system offers you the Kanji character(s) and waits for a decision of the user.

The NeXT computer, which was japanized by Canon, uses a " Kana Kanji Conversion Server " which runs as a separate process in the background. The JIM from NeXT computer (Canon) got several different modes :

- Romaji (ASCII) input
- Hiragana input (via Romaji or Hiragana keyboard)
- Katakana input
- Conversion to Kanji characters

If the JIM is switched to the " normal " ASCII input mode it works like a ASCII Keyboard, i.e., it is passing the input through to the application.

If the mode is switched to the Hiragana mode the JIM will try to interpret all key-strokes as Hiragana. First the entered character is displayed in Romaji. If a single character represents a Hiragana syllable (like a, i, e, o ,u) it will be displayed on the spot as Hiragana. If there are several choices (like sa, se, si, so, su) the JIM will display the first Romanji and wait until it is possible to decided which Hiragana it meant (see figure 4.40, page 127). Then the appropriate Hiragana will be displayed (instead of the Romaji characters, see picture a, figure 4.39, page 126). In the Katakana mode the system will do excatly the same, only that now Katakana is selected and displayed (see picture b, figure 4.39, page 126). If the computer is connected to a Hiragana keyboard the minor difference is that now there is only one key to press, which represents a Hiragana character.

This mode of conversion has the advantage that a user could easily enter the Yomi of a Kanji. Then let this input convert to the appropriate Kanji characters by the conversion routine. However, as you maybe have recognized, you have to know the correct Yomi of the Kanji which you have in mind (like "Tokyo" is spelled "Toukyou" in Japanese). This leaves us with the problem that the Japanese know and use different systems for transliterating their Yomi to Romaji. As an example you will find a table with the Hepburn and Nippon-siki (Japanese system) starting from page 69.

### Entering Katakana

The other Japanese syllable alphabet, Katakana, which is mainly used to describe foreign words or terms is entered in the same way as Hiragana. After switching to the Romaji to Katakana conversion mode it is possible to type in the syllable on the computer keyboard (see picture b, figure 4.39, page 126 or picture d in figure 4.38 on page 125).

Katakana characters are usually used to write foreign word (like names) or so called Gairaigo (loan-word from other languages). A foreign name like Schilke (spelled as Shi<sup>3</sup> ru<sup>4</sup> ke) would look like picture e, figure 4.38 on page 125.

The spelling in Katakana depends on the pronunciation of the foreign word. Following the pronunciation, the word will be represented in the "best fitting" spelling, which is possible with the Katakana syllables. Sometimes foreign words become a different spelling and pronunciation compared with the original spelling and pronunciation, e.g., the word "software" will become "sofutoeā" which looks in Katakana like picture f, figure 4.38 on page 125.

Besides these methods some systems also offer a way to enter a number code which represents the appropriate Kana or Kanji character. For the code number is the code of the character, e.g., in the system internal code, the standard JIS Ku-ten code, the JIS code or the IBM PC code (see figure 4.42 at page 129). This is only a short list, depending on the system there will be some other codes available (like Shift-JIS, ...).

---

<sup>3</sup>sounds like the SCHI*lke* in my name

<sup>4</sup>There is no L available in the Kana syllable alphabet, so that the Japanese us RU instead of L

Old Japanese Typewriter



Figure 4.34: Photos from the "Deutsches Museum Muenchen"



Input	Display	After Conversion
tou	とう	東
kyou	きょう	京
tou + kyou	=	東京

## A) Tan Kanji (single Kanji) conversion

Input	Display	After Conversion
toukyou	とうきょう	東京

## B) Jukugo (Compound) conversion

Input	Display	After Conversion
toukyouha、 subarashiima chidesu。	とうきょう は、すばらし いまちです。	東京は、素晴 らしい街で す。

## C) Renbunsetsu (Phrase) conversion

Figure 4.36: FEP conversion development





Figure 4.37: Japanese and German IBM PC55

- a) 1st Choice Kanji for Meiji 明示
- b) All Other Choices for Meiji 明示、明治、名辞
- c) bi, hi, ni, pi, ka, jitsu, nichi, nitsu, tachi, → 日  
 び、ひ、に、ぴ、か、じつ、にち、につ、たち → 日
- 1 Kanji with 9 ways of Yomi

- d) Tokyo in Katakana

Input Romaji	to	u	kyo	u	Kanji
Display Katakana	ト	ウ	キョ	ウ	東京

- e) Schilke (Shiruke) in Katakana

Input Romaji	shi	ru	ke	Kanji
Display Katakana	シ	ル	ケ	知家

- f) Software (Sofutouea) in Katakana

Input Romaji	so	fu	to	u	e	a
Display Katakana	ソ	フ	ト	ウ	エ	ア

Loan word (Gairaigo 外来語)

Figure 4.38: Kanji and Katakana characters

a)

Input ASCII	JIM	Display
H	H	H
Hi	<u>Hi</u>	ひ
Hir	<u>Hir</u>	ひr
Hira	<u>Hira</u>	ひら
Hirag	<u>Hirag</u>	ひらg
Hiraga	<u>Hiraga</u>	ひらが
Hiragan	<u>Hiragan</u>	ひらがn
Hiragana	<u>Hiragana</u>	ひらがな
Conversion	<u>Hiragana</u>	平仮名(Kanji)

b)

Input ASCII	JIM	Display
K	K	K
Ka	<u>Ka</u>	カ
Kat	<u>Kat</u>	カt
Kata	<u>Kata</u>	カタ
Katak	<u>Katak</u>	カタk
Kataka	<u>Kataka</u>	カタカ
Katakan	<u>Katakan</u>	カタカn
Katakana	<u>Katakana</u>	カタカナ

Figure 4.39: JIM conversion

Romaji ローマ字	Katakana カタカナ	Hiragana ひらがな
a	ア	あ
i	イ	い
u	ウ	う
e	エ	え
o	オ	お
ka	カ	か
ki	キ	き
ku	ク	く
ke	ケ	け
ko	コ	こ
sa	サ	さ
shi	シ	し
su	ス	す
se	セ	せ
so	ソ	そ
ta	タ	た
chi	チ	ち
tsu	ツ	つ
te	テ	て
to	ト	と
na	ナ	な
ni	ニ	に
nu	ヌ	ぬ
ne	ネ	ね
no	ノ	の

Figure 4.40: Kana Characters (Part 1)

Romaji ローマ字	Katakana カタカナ	Hiragana ひらがな
ha	ハ	は
hi	ヒ	ひ
hu	フ	ふ
he	ヘ	へ
ho	ホ	ほ
ma	マ	ま
mi	ミ	み
mu	ム	む
me	メ	め
mo	モ	も
ya	ヤ	や
yu	ユ	ゆ
yo	ヨ	よ
ra	ラ	ら
ri	リ	り
ru	ル	る
re	レ	れ
ro	ロ	ろ
wa	ワ	わ
wo	ヲ	を
n	ン	ん

Figure 4.41: Kana Characters (Part 2)

Input Method	Input	Display	After Conversion	Meaning
Katakana to Kanji	ヘンカン (henkan)	ヘンカン	変換	Conversion
Hiragana to Kanji	へんかん (henkan)	へんかん	変換	Conversion
Romaji to Kanji	kanji	kanji	漢字	Kanji
Internal Code Input	xb5ad	xb5ad	記	Notes
JIS Ku-ten Code Input	jb5ad	jb5ad	点	Point
JIS Code Code Input	1706	1706	右	Right
PC Code Code Input	8EFA	8EFA	囚	Prisoner

Different Input Methods for Kanji Conversion

Figure 4.42: Different Input Methods

## 4.6 Minor Cultural Differences

The highest stage of Japanization is to support all the smaller cultural differences. On the next couple of pages you will find some of the Japanese minor differences. It is not very important to support all of this, but the support of cultural depending things could make the point which let a Japanese customer chose a foreign software product.

### 4.6.1 Writing Style Specialities

The Japanese can use different writing directions (see figure 4.43). A common used style of writing is the Western writing style. Additional to this writing style there are some other writing styles used. The most common way of writing in Japan is still the vertical writing direction but also Western style writing is quite often used. Vertical and right-to-left writing is not a high demand for japanized versions of foreign software. Nowadays the Japanese get used to the Western style of writing. Right-to-left writing was used in ancient times (until WW2) and today there is not really a demand for this writing style. It would be a nice feature, but it is usual not needed. The Japanese vertical style which writes from the left to the right is also very rarely used today.

Today the Japanese vertical writing style (from the right to the left) is mainly used for books and magazines (see figure 4.43). Only if you plan to sell a desktop publishing program you have to consider to enable your program for vertical writing. Besides the vertical writing the Japanese style of writing has some other differences to the Western style of writing. The page order is an other difference to Western styled books or publications (see 4.44, 4.45, 4.46). So that Japanese open their books at the, for Westerners, wrong end. They read it from the back cover to the front cover. This also causes that the page numbering is, for Westerners, in reverse order. To show you an example you will see in figure 4.47 on page 147 a Japanese poem in Western style writing (picture A) and in picture B the same poem in Japanese writing style. The Japanese writing style implicate some other Japanese specialities, like Keisen, Kinsoku Shori, Kansuji, Hankaku and Zenkaku Characters, Rubi (top and bottom rubi, e.g., also called furigana), Amikake, Kinto-waritsuke, Japanese Punctuation, small Kanas, the Japanese format for addresses and telephone numbers. In addition

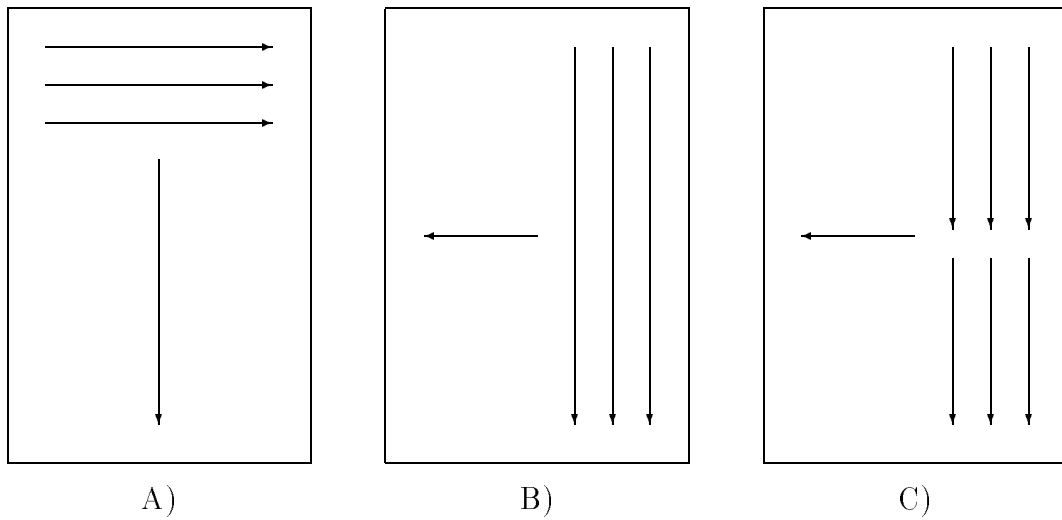


Figure 4.43: Japanese Writing Directions :

A) Western writing style (from left to the right, from top of the page to the end of the page)

B) Japanese vertical writing style (from the top of the page to their bottom, from the right side of the page to the left side)

C) Like B) with paragraphs

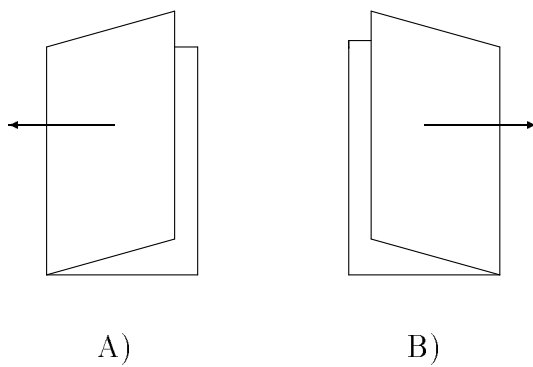


Figure 4.44: Japanese Page Order:

A) Western page order, the publication is opened from the right side

B) Japanese page order, the publication is opened from the left side



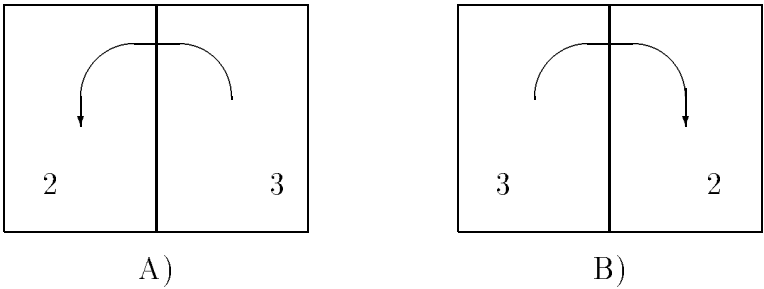


Figure 4.45: Japanese Page Order:

- A) Western page order, the publication is leafed through by turning over the right page
- B) Japanese page order, the publication is leafed through by turning over the left page

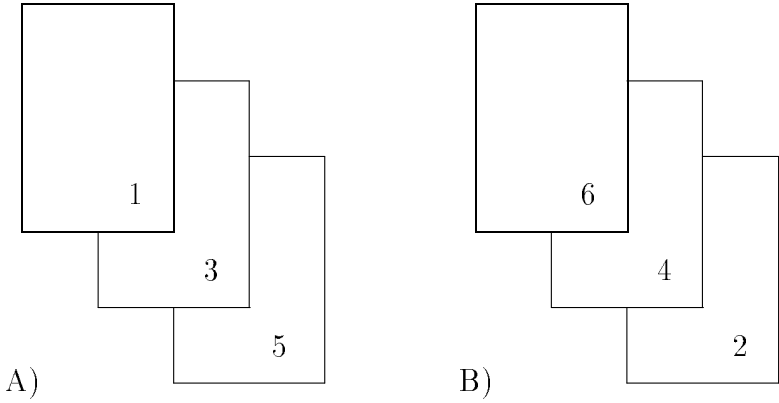


Figure 4.46: Japanese Page Order:

- A) Western page order, the pages are sorted 1,2,3,4,5,6,...
- B) Japanese page order, the pages are sorted (for us in reverse order) ...,6,5,4,3,2,1

there is a difference in the use (or understanding) of symbols for yes & no.

### **Keisen**

Keisen are used for creating tables, lists or reports. Keisen are, literally translated, lines, boxes or borders. Vertical lines are called Tatesen and horizontal lines are called Yokosen. Keisen are not a requirement for all programs. When your program produces output of data in form of tables, lists or reports it will be considered as necessary to provide Keisen.

### **Kinsoku Shori**

Kinsoku Shori is similar to the word wrap in Western word processors. It adjusts text that specific characters do not appear at the start (comma, Japanese period or symbols, see picture C in figure 4.48, page 148) or at the end of a line (open parentheses and quotation marks, see picture C in figure 4.49, page 149). This is called line head Kinsoku processing (see pictures A & B in figure 4.48) and line end Kinsoku processing (see pictures A & B in figure 4.49). Usually these characters are moved to the previous or next line.

### **Kansuji**

Kansuji is the traditional Japanese way of writing numbers. Besides the (adapted) way of writing numbers in Western style (like 1, 12, 123, 1234, ...) the Japanese use their traditional way of writing numbers. Especially when the numbers or numbers and text are written vertical Kansuji is used. In addition to "normal" numbers the Japanese use special Kansuji to indicate the numbers 10, 100, 1,000, 10,000, 100,000,000 and 1,000,000,000. On page 103 in figure 4.27 you will see the Kansuji and an example of their use. The only difference between writing numbers in Kansuji and Kanji is that the numbers 0, 1, 2, 3 have a different Kanji.

### **Hankaku and Zenkaku Characters**

The terms Hankaku and Zenkaku refer to the width of characters. For example Katakana is not always displayed in Zenkaku, which is the doubled width of normal

ASCII characters. If Katakana is displayed in Hankaku it needs the same space on the display as a standard ASCII character needs. For example on an AX-PC the size of a Hankaku character is  $8 \times 19$  dots and the Zenkaku characters is  $16 \times 19$  (in a  $24 \times 24$  matrix) dots in size. Only ASCII and Katakana are displayed in Hankaku. Depending on the code set it is represented by one or two bytes.

### **Rubi**

Rubi is another Japanese specialty. Rubi's (sometimes called furigana) are used to write the pronunciation (Yomi) of a rarely used Kanji on top (top Rubi, see picture A in figure 4.50 on page 150) or underneath (bottom rubi, see picture B in figure 4.50) this Kanji. They have usually the half size of Zenkaku characters .

### **Amikake**

Amikake is used like boldface, capitalizing or underlining text. It is somewhat like gray screening of text or characters (see picture C in figure 4.50 on page 150).

### **Small Kanas**

Small Kana characters are used to distinguish between different Yomi's of Kana characters. The character looks like a normal Kana, it is just smaller. Added to a normal Kana character it changes the Yomi of this character, e.g., Do becomes Dyo by adding a small Kana character (see picture D in figure 4.50 on page 150).

The other special Kana characters are called Dakuon, Yo'on, Sokuon and Handakuon. These are Kana characters which express different pronunciations of the standard Kana character. Dakuon are distinguished from normal Kanas by the Japanese version of the ". These characters are called "voiced sounds" (see figure 4.51 on page 151). Yo'on is a, so called, "contracted sound" and is also expressed by small Kana characters (see picture A in figure 4.52 on page 152). Sokuon is actually only one character (tsu) which is a small character and stands for a "double consonant (or assimilated) sound" (see picture B in figure 4.52). The last group of special Kana characters is the Handakuon group. This group has the Japanese ° as sign. All Handakuon start with " P " (see picture C in figure 4.52) and therefor they are called P-sounds.

**Kinto-waritsuke**

Kinto-waritsuke is a special way of formatting Japanese characters or sentences between two points. It is used to stretch text between, e.g., the beginning of a line and the end of a line (see pictures A & B in figure 4.53 on page 153).

**Japanese Punctuation**

The Japanese use several types of punctuation which are not common or unknown in the Western world. Some of them just look different from their Western counterpart. There are several different types (see picture C in figure 4.53, page 153) :

- the Japanese period
- the Japanese comma
- the Japanese separator
- the Japanese repeat symbol
- other punctuations like Japanese parentheses, brackets, ...

Besides the separator and the repeat symbol they are used like Western cultures would use this punctuation. The main difference is the special shape.

The separator is used to make it easier to understand compounds of Kanji's so that it is easier to understand the meaning of a certain Kanji group. The repeat symbol indicates that the previous Kanji has to be repeated.

**Address Format**

The Japanese address format is totally different from the address formats used in the Western societies. The way of writing an address in America would look like this :

Addressee's Name

Number Street

City State (or State abbreviation) Zip Code

An address on a German letter would look like this :

Addressee's Name

Street    Number

W- (or O- for the former East German area) Zip Code    City

The Japanese use a reverse order compared to this two Western styles :

Postal Symbol    Zip Code

City ( Town or Village)

District    Number

Addressee's Name

If your program prints anything with an address on it (invoices, letters, ...) you have to enable your program to cope with this style of Japanese Address.

### Telephone Numbers

Phone numbers are, like the date, written in various formats. In addition to the western way of writing phone number (grouped in 3 digit or 4 digit groups) like :

- (03) 3479-2893
- 03-3479-2893

there is a way of writing the number in vertical writing as you will see in figure 4.54 on page 154. As well as the use of Kansuji it is also common to write a phone number vertical with western numbers.

### Yes / No

An interesting difference between Western culture and Japanese culture is the way of expressing Yes and No. For a person with a Western background an **X** usually represents Yes and a circle ( ○ ) represents NO. In Japan these symbols will be

Product	CGA	Hercules	VGA
Chicago Software	○	○	X
Dallas Software	○	X	X

Table 4.8: Demo Table for Japanese Yes / No

interpreted in the opposite way. In a Japanese Table ○ represent Yes and the **X** represents NO. As an example let us have a look at table 4.8. An European or American reader would interpret this table in the way that Chicago Software supports only VGA and Dallas Software supports VGA and Hercules. A Japanese reader, instead, would assume that Chicago Software supports CGA and Hercules and Dallas Software only CGA.

In addition the way of answering is also different. If the system asks you " Don't you want to delete this file ? Yes / No " a Japanese user would answer " **YES**, I don't want to delete it " ([11])

### 4.6.2 Other Differences

Further to the differences mentioned above there are plenty of other differences in the Japanese (language & cultural) computer environment. I can not list all of them but I will give you some other differences which could apply in a software product.

For example : if you want to adapt a CAD program for the Japanese market or if your program just has to make some paper output you should know which units and which paper size is used in Japan. Sometimes it is useful to know which differences between foreign and Japanese hardware exists. Or think about the problem how to sort Japanese data. Even if you do not need to sort Japanese data you need at least a data-type to store the data.

#### Units

The Japanese use, unlike the English or American, mainly the metric system ([4]). To express a length they use meter as a base unit (also depending forms like millimeter,

centimeter, kilometer, etc.) and the cubic or square form of the base unit for area and volume (see tables a, b & c in figure 4.55 on page 155).

Another form to express a volume is the base unit liter. For the measurement of weight the base unit gramme is used. The base unit for time is the second. To represent a time hour, minute and second are used (see tables a & b in figure 4.56 on page 156). For temperature the Celsius scale is used ([4]).

Maybe you have recognized the "mainly" above. This implies that the Japanese use not only the units from the metric system. In addition to metric system the Japanese use some of their traditional units for measurement purposes. You will find them in the figures 4.55 and 4.56. For example real estate agents often use the traditional measures for area to describe the size of an apartment or room (e.g., Tsubo).

### **Paper size**

Similar to the unit system, the Japanese use the same paper sizes as, e.g., Australia, Germany, etc. In table 4.9 you will find the sizes of the standard which is used in Japan. This standard corresponds, e.g., to the German DIN 66008. Additionally you will find some of the American paper sizes (letter and Legal).

The most common used paper sizes in Japan are A4, B4, B5 and A3. For normal business communication A4 is mainly used. Some governmental forms are printed on B sizes.

### **4.6.3 Hardware**

Regarding the fact that a Japanese computer user has several special requirements it is naturally that there are some differences in the hardware. The major differences are caused by the fact that the Japanese use Kanji characters. This causes certain requirements, like :

- Kanji ROMs for the Kanji fonts & character set (nowadays sometimes realized as softROMs)
- high resolution for the output on screen and/or printer usually a Kanji character is coded in a 24×24 matrix (this was the main reason for the development of the 24 (!) wire printer and laser printers).

Size	JIS (DIN) A †	JIS (DIN) B †	JIS (DIN) C †
0	$841 \times 1189$	$1000 \times 1414$	$917 \times 1297$
1	$594 \times 1189$	$707 \times 1414$	$648 \times 917$
2	$420 \times 594$	$500 \times 707$	$548 \times 648$
3	$297 \times 420$	$353 \times 500$	$324 \times 458$
4	$210 \times 297$	$250 \times 353$	$229 \times 324$
5	$148 \times 210$	$176 \times 250$	$162 \times 229$
US Letter	$215 \times 279$ †		
US Legal	$215 \times 355$ †		

Table 4.9: Standard Paper Sizes used in Japan

† = in mm

Source : [1], 7-1

- a keyboard which supports Kanji character input with special conversion keys (not really necessary, but more convenient for the user)
- support of the JIS Kanji character set standard, e.g., the most printers work with this standard (as control language the Japanese version of the Epson ESC/P printer control language is widely used)
- in the Japanese PC world exists a different disk format in addition to the IBM PC standard formats. This format is 1.2 MB on a 3.5 inch floppy disk (Disk-drives which supports this format must be able to change the speed of the disk-spin).
- the size of a computer system is more important in Japan than it is in Europe or the USA. As mentioned before, the office (and private) space in Japan is limited. This creates a demand for small, but powerful, computer systems (for a picture of a compact office computer system see figure 4.57 on page 157).

These are only the obvious differences, but it gives you a fair impression about it. In the workstation world the differences are not so big and mostly solved by the use of software (like softROMs, softfonts, ...)



Let me use as an example the, so called standard in the western computer environment, the IBM PC (or compatible). Compared with the NEC PC98XX series, the market leader in Japan, you will spot some significant differences ([22]). Both machines run under the operating system MS-DOS (or PC-DOS). A "clean"<sup>5</sup> DOS program which uses only DOS calls will probably run and also "clean" Windows programs are usually compatible.

If a program relies on PC BIOS<sup>6</sup> calls the it will not run on a NEC PC. In addition the DOS of the IBM PC is a standard ASCII (IBMSCII) OS which supports only some European character extensions. The NEC PC instead runs a full Kanji version (using Shift JIS) of MS-DOS.

The most differences are based in the different hardware design of the NEC PC. These differences are :

- Keyboard layout, in order to make it easier for the Japanese user NEC added some special keys for the use with the FEP, some special application function keys and renamed some other keys. Another difference is that the NEC PC uses a FEP to handle the Kanji input (compared to the US IBM PC). Starting from page 157 you will find some examples for different Japanese keyboard layouts.
- Floppy disks are compatible in some way. The NEC PC can read the IBM PC disks in the formats 360KB (5.25"), 1.2MB (5.25") and 720KB (3.5"). In addition the NEC is able to write IBM PC disks in the 1.2MB and 720KB formats. It is not possible for the NEC to access the 1.44MB (3.5") formatted disks or to write on a 360KB floppy disk. The format which is only supported by Japanese computers is the 1.2MB (3.5") format. The IBM PC is not able to read or write this Japanese format because the system must be able to switch the speed of the disk spin.
- The Video memory has a totally different design. Instead of one byte for a character the NEC always uses two bytes. Not only the way of storing a character is different also the display attributes have a different organization then the IBM PC display attributes.

---

<sup>5</sup>using only the specified DOS calls or Windows APIs

<sup>6</sup>Basic Input Output System

- When NEC launched the NEC PC it was a requirement to work with Kanji characters. To do this they needed a higher resolution than the IBM PC was offering at this time. The NEC PC resolution in normal mode is  $640 \times 400$  pixel and the high resolution mode it is  $1120 \times 750$  pixel. This specification has not changed in the last ten (!) years.
- The NEC PC Kanji fonts are stored in a ROM chip. Nowadays the IBM PC stores them in the PC memory.
- The BIOS calls are the main hurdle for foreign software or programmers. In table 4.10 (on page 141) you will see the main differences between the IBM PC BIOS and the NEC PC BIOS. In addition the BIOS RAM area (which keeps

IBM PC BIOS	BIOS Call	NEC PC BIOS
$09_{Hex}$ & $16_{Hex}$	Keyboard	$18_{Hex}$
$10_{Hex}$	Video	$18_{Hex}$
$13_{Hex}$	Disk	$1B_{Hex}$
$14_{Hex}$	Serial Communications	$19_{Hex}$
$15_{Hex}$	System Services	various
$17_{Hex}$	Printer	$1A_{Hex}$

Table 4.10: IBM vs. NEC BIOS calls

([22])

the system status, keyboard buffer, information about the graphics mode) has a different layout.

These are the main differences. On the Japanese market there are many other players which have their own hardware and software design (like Fujitsu, Toshiba, ...).

In the workstation environment the differences are not so big. The most differences could be handled by software. The main difference in the hardware is the Japanese keyboard.

#### 4.6.4 Sorting

It is quite easy to sort data containing alphabetic or numeric data. Even if you have special characters like German umlauts ( e.g., the "a is treated as ae, it is possible to use a normal alphanumeric sort routine). For the most European or alphabet oriented countries there is a definition for the collating sequence, but now ask yourself : How would you sort pictures ? In an easy way you could look at a ideographic Kanji character as a picture. Now you should be able to imagine how difficult it is to sort Japanese data. There are several approaches to sort Japanese data, like :

Through the fact that you could compare Hiragana and Katakana with an alphabet (a syllable alphabet) it would be possible to sort data using the Yomi of the, e.g., name, instead of the ideographic Kanji characters which represent the name. To do this you have to store for each data-field (in a separate field) the pronunciation. The storing of the Yomi is necessary because a Kanji character has not only one, sometimes several, Yomi's. The pronunciation also depends on the way of reading the Kanji character. The most common ways are On-yomi (Chinese) and Kun-yomi (Japanese) reading (or pronunciation) This makes it much easier to sort Japanese data, but the problems just have begun. The main problem is that there is no Japanese standard ordering (or collating sequence) defined. This leaves us, again, with the problem how to sort Japanese data.

To sort data represented in Hiragana characters you have to use a collating sequence like the 50-On-Sequence (starting a-i-u-e-o (instead of a-i-e-o-u in the western world), called Gojuuonjun) or the I-RO-HA collating sequence (called after an old Japanese poem. It starts I-RO-HA and all Hiragana characters appear just once). A table (see figure 4.60) with these collating sequences starts from page 160. By using one of these sort sequences you also have to distinguish between sorting after the actual Yomi, the On-yomi or Kun-yomi. Furthermore this way of sorting Japanese data, there are several other ways of doing this. The disadvantage of the method described above is that you always have to store the Yomi for the sorting process. If you want to avoid this you have to use one of the other methods of sorting Japanese data.

Other ways of sorting Japanese data are, e.g., sorting after the numbers of strokes of a Kanji character (this collating sequence is called Sokaku), after the type of the strokes (called Bushu) or after the JIS code table (which is actually the easiest way of sorting Japanese data).

Sorting after the JIS code table brings the data in the following order ([26], [27]) :

- JIS X0208 level 1 is sorted after the representative On-Kun Yomi of the Kanji character by using the 50-On sequence. Kanji characters with the same Yomi are sorted in the order On-yomi, Kun-yomi, order of Radicals (see below), number of strokes.
- JIS X0208 level 2 is collated in the 214 classes of Radicals (see below). In these classes the Kanji characters are sorted after the number of strokes. If some characters have the same number of strokes they are sorted following the 50-On sequence.
- JIS X0212 (sometimes called JIS level 3) is sorted like JIS X0208 level 2 in the 214 groups of Radicals and within a Radical group after the number of strokes.

In the description of the collating sequence which is used for the JIS character set we find a new term called Radical. A Radical (part-head group) is an ideograph, a base which is used in combination with other Radicals to form Kanji characters. Usually a Kanji character is formed by up to four Radicals. On page 162, picture a in figure 4.62 you will find some Kanji characters which are formed with one (No. 1), two (No. 2), three (No. 3) and four Radicals (No. 4). There are several ways of combining Radicals to a Kanji character, e.g., □, □□, □□□,  $\begin{smallmatrix} \square \\ \square \end{smallmatrix}$ ,  $\begin{smallmatrix} \square & \square \\ \square & \square \end{smallmatrix}$ , ... (with □ representing a Radical).

So you see that it is not easy to sort Japanese data, but there is a way to do this. If you do not want to implement your own sort algorithm you could by a package which does this for you.

#### 4.6.5 Japanese Data-types

In order to enable a programming language (or application) to work with Japanese characters we have to enable the system to work with DBCS characters. There are two ways of enabling a programming language or application to cope with DBCS characters.

1. Using the old data-types and enabling the system to work with DBCS characters. The Japanese version of Oracle goes this way. If you define a data-field of

20 character it is a normal SBCS data-field which could store up to 20 SBCS character. By enabling the system to work with DBCS character it is now able to store up to 10 DBCS characters or a mixed string containing any combination of SBCS and DBCS characters. In the case of a mixed string the byte length of the string can not exceed the limit of 20 byte.

2. By defining a new data-type for the handling of DBCS character (or mixed) strings. This is done, e.g., for COBOL (DISPLAY-2), FORTRAN (NCHARACTER) and C (wchar\_t) ([9]). The problem with the extended data-type is that the compiler has to be rewritten because all functions handling character data have to work character oriented instead of byte oriented. The new data-type affects all types of commands, like input/output, assignment, string handling, sort/merge and comparative operations.

There are two ways of implementing a DBCS data-type. For the example I will use the widely used programming language C. The ISO 9899:1990 (ISO C) standard ([5]) defines the C data-type wchar\_t. This data-type is a wide character data-type and stores the character data like 8C8E<sub>Hex</sub>. In addition this standard defines routines for the conversion between multi-byte and wide characters. The second approach is to store data as a multi-byte character which looks like 8C<sub>Hex</sub>8E<sub>Hex</sub>. In this approach n single bytes are used to store the information. As a wide character the data is stored as a group of n-bytes (8C8E<sub>Hex</sub> vs. 8C<sub>Hex</sub>8E<sub>Hex</sub>, [6])

The advantage of a programming environment which can handle DBCS characters is that the programmer is able to write internationalized programs. The program is able to run in different national computer environments. The emerging problem is that the programming environment has to be rewritten that it is able to handle (probably different) DBCS character sets. In addition the system must offer different conversion routines between SBCS and different DBCS (also between multi-byte characters and wide characters)

#### 4.6.6 Japanization Pitfalls

In this paragraph I will talk about some pitfalls which could cause misunderstanding if you are not careful enough when you japanize your product.

One of the common anecdotes about a japanized product is the story about the beep in the Japanese version of Lotus 123. As mentioned before office space is rare in Japan. Many people work in open-plan offices. When Lotus launched their first version of Lotus 123J they discovered that they had to remove the error-beep. First of all it is very shameful for a Japanese when the computer tells everybody " BEEP, you made a mistake " and in open-plan offices the steady beeping would cause a major disturbance. As mentioned in the date section the Japanese use a special date format which contains the era of the emperor. In one of the earlier Lotus 123 versions it was possible to add the common Gengou date or the reign of the emperor and it was possible to change the name of the emperor. This was a big mistake because it looked for the Japanese that one was planning the death of the emperor ([12]).

Two other things, from the thousands of pitfalls, I will mention here are translation errors and manual design. When the first translation of the UNIX operating system took place there were some wrong translations made when they adapted UNIX to the Japanese language environment. The development of UNIX mainly took place in the US academic environment, so many UNIX-related terms have funny names (for US people) like demons, zombie process and killing a (child) process. In the first translation the message " a child process was killed " was translated to the horrifying Japanese sentence " we just murder your first-born child ".

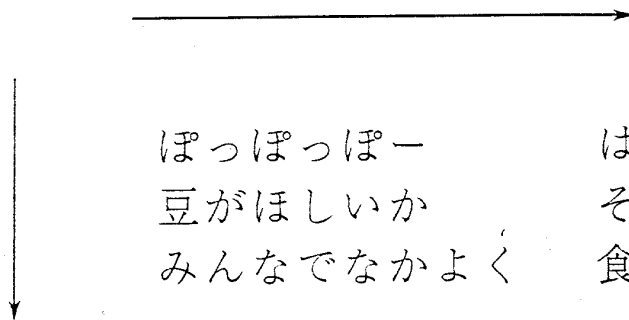
The Japanese use a lot of adapted English technical denotation but sometimes they prefer more (for them) visible Japanese terms, e.g., a Kanji which expresses an idea better or more understandable for Japanese, then the English loan-word. If you let translate the manual you should always use a Japanese native speaker for the translation. After the first translation is done you should let a second technical translator do a back translation, to check the translation of the first translator ([20]).

This is important because now you can control if the first translator gets the point which you want to express in the manual. Another fact is that the Japanese manuals have a different style than US or European manuals. In Japan the manual describes a scenario and tries to explain the user (with examples) how to work with the product. Besides that small cartoons are very common, like a floppy disk which tries to avoid the contact with a magnet (see on the cover of the most  $5\frac{1}{4}$  floppy disks).

### 4.6.7 Cultural Differences

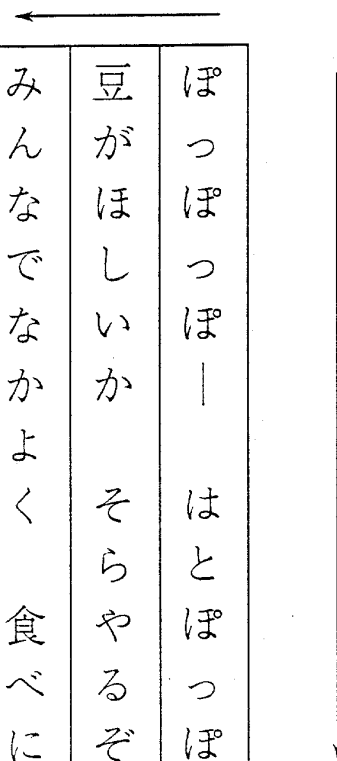
You have read now a lot about Japanese cultural differences. Not all of them have to apply for an adapted program. Nevertheless always some of them apply. If you do the japanization for your program you have to think about which of these differences you have to adapt and which not. You could start by adapting the, for your users, more important features and later on you adapt more and more of the cultural differences.

If you think you do not have adapt even some of the minor differences you will recognize that your software will not sell well (except you have a market-niche). It is like that you are offering a word-processor in Germany which is not able to handle the German umlauts. It will not be a good seller.



ぽっぽっぽー	はとぽっぽー
豆がほしいか	そらやるぞ
みんなでなかよく	食べに來い

A) Japanese poem in Western horizontal writing style



み	豆	ぽ
ん	が	っ
な	ほ	ぽ
で	し	っ
な	い	ぽ
か	か	ー
よ	そ	は
く	ら	と
食	や	ぽ
べ	る	っ
に	ぞ	ぽ
來		ー
い		

B) Japanese poem in Japanese vertical writing style

Figure 4.47:



.... AX-1とAX-2があります  
 。 AX-2はスーパーインポーズができますが、AX-1では.....

A) Text before Kinsoku Line Head Processing

.... AX-1とAX-2があります。  
 AX-2はスーパーインポーズができますが、AX-1では.....

B) Text after Kinsoku Line Head Processing

Symbol	Meaning
。	Japanese period
、	Japanese comma
,	Comma
.	Period
.	Raised dot
!	Exclamation mark
?	Question mark
:	Colon
;	Semicolon
-	Hyphen
々	Character repeat symbol
)	Close parentheses
>	
}	
」	
'	Quotation marks
”	
イ、エ、...	Lower case katakana used for loan words

C) Prohibited Symbols at line head

Figure 4.48:

．．． AXマシンには AX-1とAX-2 (スーパーインポーズ可能)の2つのタイプがあります。

A) Text before Kinsoku Line end Processing

．．． AXマシンには AX-1とAX-2 (スーパーインポーズ可能)の2つのタイプがあります。

B) Text after Kinsoku Line end Processing

Symbol	Meaning
(	Open parentheses
< ≪	
[ {	
{	
「 『	Quotation marks
,	
”	

C) Prohibited Symbols at line end

Figure 4.49:

あめ  
雨

水  
みず

A) Top Rubi

B) Bottom Rubi

これは、あみかけです。

C) Amikake

Romaji	Kana	Romaji	Kana
Do	ど	Dyo	ぢょ
Re	れ	Rye	りえ
Mi	み	Myi	みい
Do	ド	Dyo	ヂョ
Re	レ	Rye	リエ
Mi	ミ	Myi	ミイ

D) Small Kanas

Figure 4.50:

a) Dakuon      だくおん      濁音

Romaji ローマ字	Katakana カタカナ	Hiragana ひらがな
ga	ガ	が
gi	ギ	ぎ
gu	グ	ぐ
ge	ゲ	げ
go	ゴ	ご
za	ザ	ざ
zi	ジ	じ
zu	ズ	ず
ze	ゼ	ぜ
zo	ゾ	ぞ
da	ダ	だ
di	ヂ	ぢ
du	ヅ	づ
de	デ	で
do	ド	ど
ba	バ	ば
bi	ビ	び
bu	ブ	ぶ
be	ベ	べ
bo	ボ	ぼ

Figure 4.51:

a) Yo'on    ようおん    拗音

Romaji ローマ字	Katakana カタカナ	Hiragana ひらがな
a	ア	あ
i	イ	い
u	ウ	う
e	エ	え
o	オ	お
ya	ヤ	や
yu	ユ	ゆ
yo	ヨ	よ

b) Sokuon    そくおん    促音

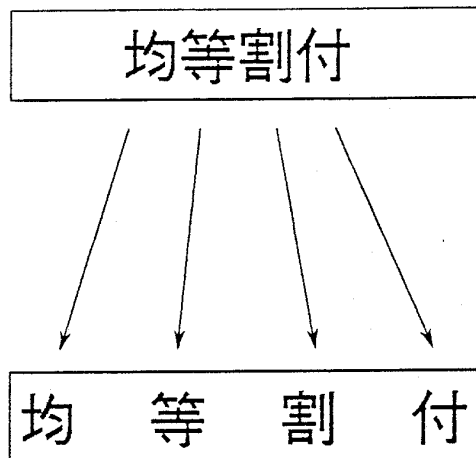
Romaji ローマ字	Katakana カタカナ	Hiragana ひらがな
tsu	ッ	っ

c) Handakuon    はんだくおん    半濁音

Romaji ローマ字	Katakana カタカナ	Hiragana ひらがな
pa	パ	ぱ
pi	ピ	ぴ
pu	プ	ぷ
pe	ペ	ぺ
po	ポ	ぽ

Figure 4.52:

A) Text before Kinto-Waritsuke Processing



B) Text after Kinto-Waritsuke Processing

Name	Symbol
Period	。
Comma	、
Separator	・
Repeat Symbol	々
Parentheses	( ) ( ) [ ]
Corner Brackets	「 」 『 』
Pointed Brackets	< > << >>
Ellipsis	...
Question Mark	?
Exclamation Point	!

C) Punctuation Symbols

Figure 4.53:

## a) Telephone Number in Japanese Writing Style

Tel (03) 3479-2893

T	電
e	話
l	
(03)	(03)
3479	3479
28	28

Fax (03) 3479-5579

F	フ
a	ァ
x	ックス
(03)	(03)
3479	3479
5579	5579

Figure 4.54:

## a) Length

Name	Unit	Sign
Meter	1m	m or 米 or メートル
Ri	3.9273km	里
Shaku	30.30cm	尺
Sun	3.03cm	寸

## b) Area

Name	Unit	Sign
Square Meter	1m <sup>2</sup>	m <sup>2</sup> or 平米 or 平方メートル
Square Kilometer	1000m <sup>2</sup>	km <sup>2</sup> or 平方キロ米 or 平方キロメートル
Jou	1.62m <sup>2</sup>	畳
Tan	991.7m <sup>2</sup>	反
Tsubo	3.306m <sup>2</sup>	坪

## c) Volume

Name	Unit	Sign
Rippou Meter	m <sup>3</sup>	m <sup>3</sup> or 立米 or 立方メートル
Liter	l	l or リットル
Milliliter	ml	ml or ミリリットル
cc	cc	cc
Gou	180.39cm <sup>3</sup>	合
Shou	1.8039l	升
To	18.039l	斗

Figure 4.55:



## a) Weight

Name	Unit	Sign
Gramme	g	g or グラム
Kilogramme	kg	kg or キログラム
Ton	t	t or ton or トン
Kan	3.75kg	貫
Momme	3.75g	匁

## b) Time

Name	Unit	Sign
Byou	Second	秒
Hun	Minute	分
Ji	Hour	時

Figure 4.56:



Figure 4.57: .



Figure 4.58: Japanese Keyboards, Page 2



Figure 4.59: Japanese Keyboards, Page 3

Romaji ローマ字	Hiragana ひらがな 50-On- Sequence	Romaji ローマ字	Hiragana ひらがな Old-Sequence
a	あ	i	い
i	い	ro	ろ
u	う	ha	は
e	え	ni	に
o	お	ho	ほ
ka	か	he	へ
ki	き	to	と
ku	く	chi	ち
ke	け	ri	り
ko	こ	nu	ぬ
sa	さ	ru	る
shi	し	wo	を
su	す	wa	わ
se	せ	ka	か
so	そ	yo	よ
ta	た	ta	た
chi	ち	re	れ
tsu	つ	so	そ
te	て	tsu	つ
to	と	ne	ね
na	な	na	な
ni	に	ra	ら
nu	ぬ	mu	む
ne	ね	u	う
no	の	wi	ゐ

Figure 4.60:

Romaji ローマ字	Hiragana ひらがな 50-On-Sequence	Romaji ローマ字	Hiragana ひらがな Old-Sequence
ha	は	no	の
hi	ひ	o	お
hu	ふ	ku	く
he	へ	ya	や
ho	ほ	ma	ま
ma	ま	ke	け
mi	み	hu	ふ
mu	む	ko	こ
me	め	e	え
mo	も	te	て
ya	や	a	あ
yu	ゆ	sa	さ
yo	よ	ki	き
ra	ら	yu	ゆ
ri	り	me	め
ru	る	mi	み
re	れ	shi	し
ro	ろ	we	ゑ
wa	わ	hi	ひ
wo	を	mo	も
n	ん	se	せ
		su	す

Figure 4.61: 50-on Sequence and I-RO-HA

a)Radicals

- 1) 木 (Ki、き、Tree)
- 2) 林 (Hayashi、はやし、Wood)
- 3) 森 (Mori、もり、Forest)
- 4) 街 (Machi、まち、Town)

Figure 4.62: Radicals

# Chapter 5

## Japanization

There are several ways and levels of japanizing a software product. Some few products sell even without been japanized (mainly computer games and niche-market products). Japanese EDP specialists are used to work with English products. However the normal Japanese user is (or will not be) satisfied by using foreign language software products.

Even if it does not look so, the entry of personal computer systems in Japanese offices was not that long ago. About five or seven years ago the most used computer system in Japan was the mini to main frame range of computers. Since the fast development of personal computers started, the smaller systems got more powerful to handle the resource swallowing Japanese features (like Kanji support, ...).

### 5.1 First Steps

The first step of Japanization is to implement the program on a Japanese hardware platform. The problem with, e.g., the Japanese PC platforms is that they all have different hardware layouts and specifications compared to an IBM PC. Only "clean" programs could be transferred and (maybe) used as a plug and play version (e.g., for a demo) in a Japanese MS-DOS environment. The usual way is to recompile the whole program with a Japanese version of the compiler. Before that you have to change all hardware depended BIOS calls and get rid of "dirty" code. You have to do this because in Japan there are about seven different PC hardware platforms. This makes



it very difficult to localize a product which peeks and pokes around in the system. If you use, e.g., undocumented MS-DOS calls you can not be sure that this call exists in the Japanese version of a hardware vendor MS-DOS version.

If you have ” cleaned up ” your program and compiled it on a Japanese hardware platform you will have an original version of your program which runs on (one) Japanese hardware platform. This version has the big disadvantage that it does not support any of the Japanese language specialities.

If it is a special product, e.g., for system administrators or other EDP specialists you maybe could sell it to Japanese customers.

### 5.1.1 Adding Japanese features

Some big international companies, like Citibank, use their original programs and add a Japanese Front End (do not mix this up with Front End Processor). This means that the program body is not changed, it can not work with Japanese Kana or Kanji characters. The new Front End provides the Japanese user just with a Japanese user interface, like Japanese menus and help-screens. The functions of the program are not changed. This is a way to make it easier for Japanese computer user to use a program in an international environment. Nevertheless this approach applies only for big international companies or programs which not have to work with Japanese data.

The most important step in japanizing a software product is to enable it to work with DBCS characters, i.e., with the Japanese syllable alphabets Hiragana, Katakana and the ideographic Kanji characters. In order to do this you could create a Japanese, a Chinese, a Korean, ...version or you enable your product to run in every national language DBCS environment (more about that topic later).

If you want to enable your software to work with DBCS character sets, i.e., you have to use data-types which are able to handle this type of characters (like the `wchar_t` data-type in C or other specific Japanese data-types provided by the compiler maker).

After doing this there are two ways to choose from. You could launch the first version of your DBCS enabled program without changing the menus and help-screens, just by adding a Japanese manual. The second, more complicated, way is, not only to enable your product to handle Japanese characters, but also to adapt the manual, menu and help system to the Japanese language. The first way could be a good choice

if you just want to be present on the Japanese market and later on (if the product seems to be successful) you want to launch a " more " japanized version.

It is clearly visible that the version with DBCS support, japanized menus and help-screens will be more successful in the Japanese marketplace. For the beginning or for niche-market products a version with DBCS support and, e.g., English menus would also sell.

### 5.1.2 Full Japanization

The last stage in japanizing a software product is to adapt all cultural Japanese specialties which apply for this product. For a word processor this would be things like Amikake, Keisen, Kinsoku Shori, Rubi characters, Kinto-waritsuke and so on. If you japanize a real estate agent program the Japanese user would appreciate if the system would support the Japanese units for measuring the floorspace (e.g., Tsubo).

It is naturally that everybody likes to work in well know environment. Imagine a Japanese user who has to work with, e.g., a program which prints the date in the US date format (MM/DD/YY). This is just a minor difference but it could cause misunderstanding. In addition the user could not use the system to prepare papers for the government because they insist on the Japanese date format.

If you support all the other small cultural differences it will help your system to compete on the Japanese software market.

### 5.1.3 Japanese Operating Systems

To give the appropriate support for the Japanese user the operating system has to support several features. First of all the Japanese user will appreciate if he could use file, volume and device names written either in Hiragana, Katakana or Kanji characters. Furthermore a complete help system and error messages in Japanese should be available.

For applications, commands and programming languages it is necessary to provide the possibility to write comments and the names of variables in Japanese. In addition it should be possible to use special Japanese data-types (for DBCS characters). A nice feature would be that reserved words could be written in Japanese.

## 5.2 Japanese Software Environment

The PC/WKS market will become the most important market in Japan. The two leading operating systems are MS-DOS and UNIX. Both are existing in japanized versions. The PC market is a very diverse market with many different hardware platforms and implementations of MS-DOS. Since the MS-Windows market share in Japan is growing the difference between the operating systems has started to fade away.

The UNIX workstation market is more homogeneous. Even if there are different hardware platforms and UNIX implementations. The different UNIX operating systems implementations, which are always licensed by the UNIX Systems Laboratory, uses all the same program interface. So if you use a UNIX system there should be only minor problems in adapting a software to this environment ( at least less then in the Japanese MS-DOS environment).

### 5.2.1 MS-DOS

The Japanese DOS market is a mess (!). There are several vendors which offer their hardware with MS-DOS. Unfortunately each platform has a different hardware layout and implementation of MS-DOS. The best example is the market leader NEC with the PC98XX series. This machine uses an architecture which is similar to the architecture of the IBM PC. Nevertheless the machines are incompatible because the engineers of NEC used, e.g., different hardware addresses for controller chips, placed other interrupt routines behind interrupts as IBM did and designed a more complex video memory design for the handling of Japanese characters. NEC is not the only vendor who did this. All of the other vendors (like Fujitsu, Toshiba, ...) did the same thing.

Moreover the mess of different implementations there are also many different FEPs for PCs available in Japan. If your program was developed by using, e.g., the VJE- $\beta$  then you can not be sure that an other FEP supports the same features as this FEP. Switching or controlling the mode of the FEP is not possible except this feature are provided by the standard FEP interface.

## Clean Programs

The only programs which should run on every MS-DOS machine are really ” clean ” programmed programs. This means that you only could use the fully documented MS-DOS calls. If you use undocumented DOS calls, BIOS calls, your system peeks & pokes around or you try to manipulated the hardware directly your system will probably not run on a Japanese MS-DOS machine. This has caused many obstacles for software developers in and outside Japan. For every hardware platform you have to code a special version or if you would use the standard MS-DOS interface your program would be slower or could not perform some special program features. This was and is the main hurdle for foreign software vendors.

## Japanese MS-DOS

The good thing about the Japanese MS-DOS is that it is fully japanized and supports many the Japanese special cultural features. The help-system provides the help-screens in Japanese (see figure 5.3 on page 176). The system allows to use file names written in Japanese (see figure 5.4 on page 179). There are many successful business applications available in japanized versions (see the picture of LOTUS 123j in figure 5.4 on page 179). Even with the bunch of hurdles and problems the system is the best choice for running PC software on Japanese PC systems.

## AX & OADG DOS

Fortunately there are two new versions of the PC operating system around which try to make it easier for the user and the software developers to run programs on different platforms. These new DOS versions are from the AX Consortium and from the OADG (Open Architecture Development Group, called DOS/V). Both DOS versions run on different platforms from different vendors. Both systems support special Japanese features like FEP, Kanji characters, . . . . The good thing about this new approach is that a program developed on an AX (or OADG) DOS machine *should* run on every other AX (or OADG) DOS machine. It is still not possible to peek and poke around or manipulate the hardware directly<sup>1</sup>, but at least all the DOS calls follow one (AX

---

<sup>1</sup>You should not do it, but AX machines have all the same hardware architecture so it should be fairly possible to run a ” dirty ” programmed AX program on every AX compatible machine

or OADG) standard. The latest developments show that both systems will become compatible to each other in the near future.

### Japanese Windows

The biggest boost for PC software compatibility, on different platforms, was the Japanese MS-Windows version 3.0 (see figure 5.3 on page 176). This version of MS-Windows was still implemented by each hardware vendor, but all implementations use the same set of windows APIs (Application Program Interface). In addition the Japanese window's version is compatible to the English (German, French, ...) window's version. The only problem with the compatibility between the different national versions is that national characters or line elements will be displayed as Katakana characters by the Japanese window's version.

This fact makes it easy to run foreign software in a Japanese computer environment. If the programmer has used the wide-character supporting windows APIs the main work of localization is already done. The only minor difference between the Japanese window's version is that each vendor has used his own technical denotation.

### 5.2.2 UNIX

In the workstation world UNIX is still the most used operating system. In Japan nearly every hardware vendor offers a UNIX based workstation line. This market is so important that SUN offers (only in Japan) a Laptop version of their SPARC based machines. This combines the power of a real UNIX workstation and the highly demanded space saving size of a Laptop.

The advantage of the UNIX operating system is that all vendors who want to implement a version of UNIX have to buy a license from the UNIX System Laboratory. This causes that all the implementations are based on the same source. In addition it is more common in the UNIX environment to have (or use) clear documented interfaces. Since a couple of years the development of the UNIX system moves towards an open system environment which relies heavily on standard communication protocols and interface descriptions.

UNIX is some kind of a layered operating system. That leads towards a simple approach of adding country specific extras. I will talk about the approach from USL

(UNIX System Laboratory) and the implementation from HP. UNIX adds to the base operation system a layer which is called Multi National Language Supplement (or MNLS, [13], [16]). This layer includes the basic functions to support different national computer environments (see figure 5.1 on page 169). This features are called

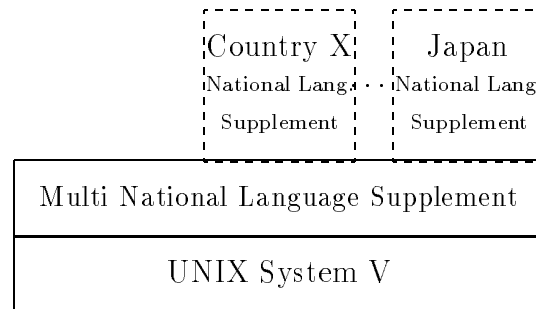


Figure 5.1: UNIX System V Layers

internationalization features and include :

- support of full 8-bit code sets. Commands are able to handle 8-bit code sets and EUC code sets.
- alternative date and time formats are supported
- enhanced support for conversion functions like upper and lower casing
- extended set of classifications for characters like alphabetic, printable, upper and lower case, ...
- ANSI C libraries which support the internationalization features like message handling, EUC<sup>2</sup> and multi-byte processing
- multi lingual message handling for retrieving messages during runtime for applications and commands
- character mapping

On top of this layer sits the Country Specific Product (called CSP). For the Japanese Unix environment exist a special CSP. This covers the main Japanese differences and

---

<sup>2</sup>Extended Unix Code

applies, e.g., for the character set mapping, FEP, .... The MNLS system of UNIX works mainly with the ANSI C compiler. The system is controlled by an environment variable called LANG. If this variable set to the Locale of a specific country (like ja\_JP.sjis or ja\_JP.euc for Japan) the system will (or should) perform all operations following the national profile for this country ([5]). Naturally it will perform this only if the program or command is written by using the localization functions of the ANSI C compiler. The locale contains the following categories (environment variables, [13]):

- LC\_CTYPE *character classification and conversion*  
specifies a different character class table (if defined)
- LC\_TIME *date and time format*  
contains the information about date and time format
- LC\_NUMERIC *Numeric representation*  
specifies the decimal point character and thousands separator
- LC\_COLLATE *collating sequence*  
holds the information about the sort order
- LC\_MONETARY *monetary information*  
information for the monetary sign, positive and negative values
- LC\_MESSAGES *language in which the messages should be retrieved*  
holds the information about the directory where the system messages will be found

Usually this categories are defined by a national body via the national country profile for UNIX. If a system command or application uses the ANSI C localization functions the system will behave following the national profile. That means that, e.g., the date or time function will deliver the output following the national specifications (standard). In addition all messages (if available) will be displayed in the national language. Country specific specialities as sort order or other conventions covered by the categories will also be performed following the national profile.

The national profile contains (or should contain) all of this country specific specialities. Once this work is done it makes it very easy to localize (or japanize) the basic

local differences in a UNIX environment. By setting the environment variable LANG from de\_DE.src (for Germany) to ja\_JP.sjis (for Japan) the representation or information about of time, date, money, numbers, messages, conversion & classification functions, sort order and character classification should change from the German definition to the Japanese ([5]).

As you may recognize there are two settings for the environment variable LANG available in Japan. The ja\_JP.sjis applies for an environment which works with the Shift JIS code table. The ja\_JP.euc applies for the more flexible EUC code table. As alluded before the ANSI C compiler supports wide-characters through the data type `wchar_t`. In addition you will find conversion & classification and string handling functions for wide-characters (and multi-byte characters).

Shift-JIS was already introduced so I will now talk about the EUC character mapping. The EUC is spilt into four groups of code sets. The code set No. 0 is always mapped to the standard SBCS ASCII character set. With the EUC character mapping it is possible to handle multi-byte character sets with a (theoretical) unlimited number of bytes (practical values are 1 & 2 bytes and for Asian languages up to four bytes, [14], [13]). To distinguish between the different code sets the EUC uses the MSB or the two single shift codes SS2 and SS3 (for the code set 2 and 3). The value of SS2 is  $8E_{Hex}$  and for SS3 is  $8F_{Hex}$  following the standards ISO 2022 and ISO 6937/3. In order to tell the system which code set is mapped to which type of multi-byte character set and the width of the characters in the set you could use a system function called `cswidth`.

The EUC uses two representations, the internal and the external code. The internal code is used to store the characters in memory. For the input / output the external code is used. These codes are called the EUC representation and the wide character representation. The wide character representation is available in a 16-bit and a 32-bit form (see the table's 5.1, 5.2 and 5.3 starting from page 177).

The advantage of this system is that it is able to handle different character sets with a theoretically unlimited number of bytes. In the tables the following examples are shown :

- one byte for the character set (a, in code set two and three with single shift character)



- two byte for the character set (b, in code set two and three with single shift character)
- three byte for the character set (c, in code set two and three with single shift character)

Furthermore the system offers a variety of multi-byte supporting functions like :

- character I/O
- string I/O
- formatted I/O
- string manipulation
- string conversion

In this environment it is possible to write truly internationalized applications. In the Japanese version the EUC is mapped to the following standard character sets (see table 5.4 on page 178). The escape sequences in this table are following the recommendation of the ISO 2022:1986 (JIS X0202-1991) standard. As you see it is possible to map all Japanese standard character sets to the EUC. This gives us a character set which could be used through out all UNIX platforms. And makes it easier to port software from one UNIX system to an other UNIX system.

Through the consistence of this approach all levels of the UNIX operating system are enabled to work in different national language environments.

## NLS

As we read does UNIX make the localization a little bit easier then it is in the MS-DOS PC environment. I want now introduce the Native Language Support (NLS) from HP. The HP UNIX supports through NLS twenty-two different native language environments. The most interesting part is that it supports all Asian languages ([17]).

In the next paragraphs I will explain the approach which HP uses. This approach is based on the MNLS features of UNIX. It not only replaces standard C routines, it also comes with a set of tools which are helpful to maintain and construct locales

and message catalogs. The approach behind the NLS is to work with a single source. If you want to produce different national (localized) versions of a program you could create for each country an own version. This was the way the most software developers have chosen in the past. Actually it is silly to work on several different sources for each supported country. You never keep track of the enhancement in all versions. The better way is to work just with one source and let the system do the localization (at runtime). As mentioned above the MNLS and the CSP allows to create a generic program. For the locale specialities the system takes care of. Naturally the system needs the country specific profile which contains the information's about the national language environment.

If you got a Japanese national profile there is no problem to run your program in a Japanese environment. In order to use one source you have to remove all strings from the program. Instead of fixed strings you use function calls which retrieve the appropriate string from a message catalog. Controlled by the LANG environment variable (and/or LC\_MESSAGES) the system uses either the default string (if the message catalog is not available) or the appropriate string from the message catalog in the national language.

Before you can use the national language message catalog you have to create one. For each language that you want to support you have to create a message catalog. For example you write the program using English default messages and later on you translate the message catalog to French, German or Japanese. This makes it very easy to localize an application because you just have to change or create a set of strings in the locale language. In addition it allows you to maintain just one source of the program. This makes it much easier for the further development and maintenance of the program.

You will find a rough skeleton for a "normal" program and a program which follows the NLS guidelines in figure 5.2 on page 174. The difference between the programs is that the "normal" program uses fixed strings for messages. The NLS program instead checks the LANG environment variable first. After that it opens the appropriate message catalog. If it is not possible to open the message catalog the NLS program uses the default strings. When the message catalog is available the system retrieves, during runtime, the messages in the language of the national environment. Before the program terminates it should close the message catalog.

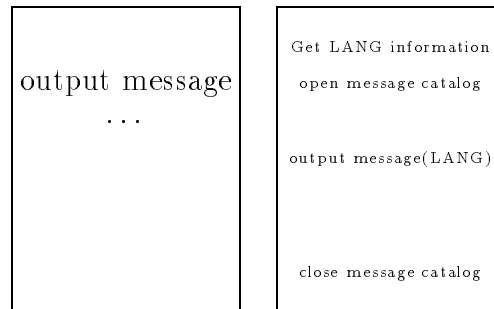


Figure 5.2: NLS program skeleton

This has the big advantage that there is only one source to maintain. To create a program for a certain national environment you need only the national profile and a message catalog in the national language.

The HP-UX NLS provides you with several tools to exclude the text strings from the program source, to create (a kind of precompiled) message catalog and several library routines for the conversion, I/O and string handling. The main part which controls the I/O for X window, terminal and printer is called NLIO (National Language Input Output). This part of the NLS provides the system with the country specific character I/O routines so that, e.g., X windows becomes capable to handle Japanese character I/O. If a certain national profile is not supported by the NLS it is possible to create a new Loacle for this language environment.

### 5.2.3 MS-DOS vs. UNIX

At the moment it looks like that UNIX has the leading edge in supporting localization. MS-DOS is widely spread in the Japanese computer world but it has many obstacles and hurdles for foreign programmers who wish to japanize software. UNIX is much more advanced in this point but it has a smaller market share then MS-DOS. In next couple of years the problems of japanizing software will not totally fade away but it will become much easier to do. Both operating systems move towards a more open and internationalized (globalized) approach. This means that in the near future, both will support localization features even in the basic version.

At the moment it is necessary to create a separate program version for each DOS

hardware platform. This could change in the near future when the new DOS versions (AX & OADG) become more widely spread.

In the UNIX world it is absolutely necessary to work out a standard for the localization of applications. The emerging open systems technology would not work between different national UNIX systems if they would not use a common scheme for the realization of local UNIX versions.

As you see it is essential to work towards a localization standard on both operating systems. The best way of doing this would be to establish a standard across all operating systems.

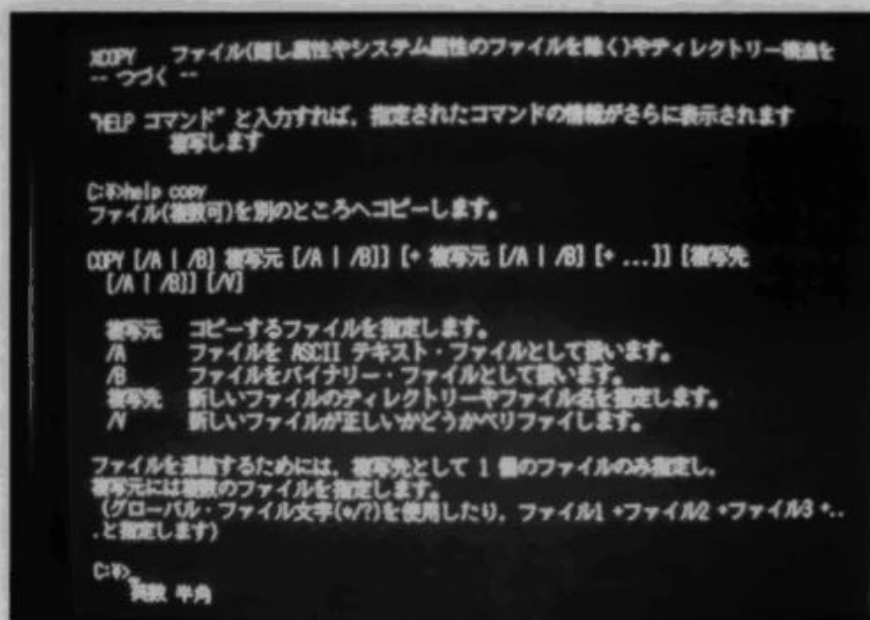
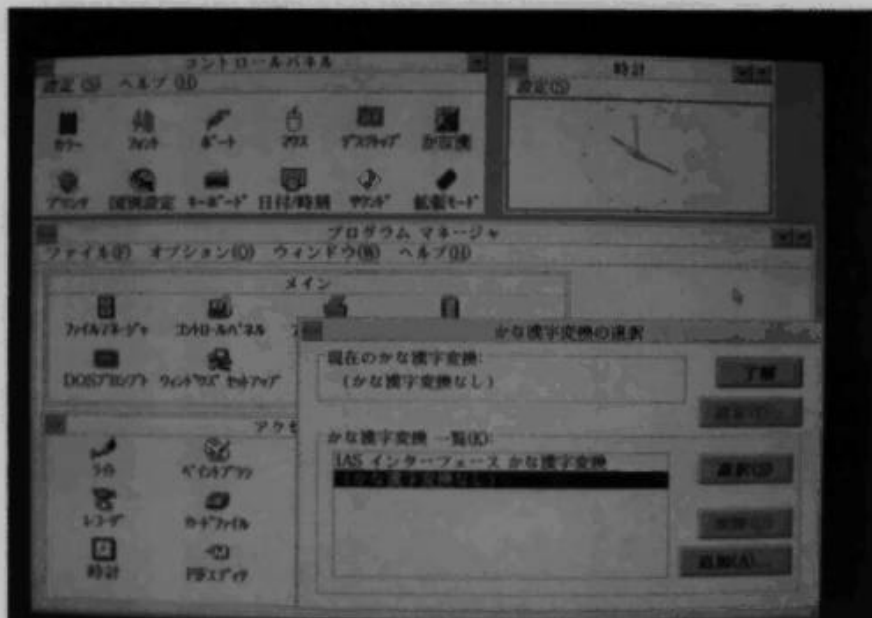


Figure 5.3: Japanese Windows and Help-screen

Set	EUC Code
0	0XXXXXXX
1 (a)	1XXXXXXX
(b)	1XXXXXXX 1XXXXXXX
(c)	1XXXXXXX 1XXXXXXX 1XXXXXXX
2 (a)	SS2 1XXXXXXX
(b)	SS2 1XXXXXXX 1XXXXXXX
(c)	SS2 1XXXXXXX 1XXXXXXX 1XXXXXXX
3 (a)	SS3 1XXXXXXX
(b)	SS3 1XXXXXXX 1XXXXXXX
(c)	SS3 1XXXXXXX 1XXXXXXX 1XXXXXXX

Table 5.1: EUC representations, External Code

Set	16-bit Process Code
0	00000000XXXXXXX
1 (a)	100000001XXXXXXX
(b)	1XXXXXXX1XXXXXXX
2 (a)	000000001XXXXXXX
(b)	0XXXXXXX1XXXXXXX
3 (a)	100000000XXXXXXX
(b)	1XXXXXXX0XXXXXXX

Table 5.2: EUC representations 16-bit, Internal Code

Set	32-bit Process Code
0	000000000000000000000000XXXXXXX
1 (a)	001100000000000000000000XXXXXXX
(b)	00110000000000000000XXXXXXX
(c)	00110000000XXXXXXX
2 (a)	000100000000000000000000XXXXXXX
(b)	00010000000000000000XXXXXXX
(c)	00010000000XXXXXXX
3 (a)	001000000000000000000000XXXXXXX
(b)	00100000000000000000XXXXXXX
(c)	00100000000XXXXXXX

Table 5.3: EUC representations 32-bit, Internal Code

Set	Escape Sequence	Character Set
0	ESC ( B or ESC ( J	ASCII (ANS X3.4-1968) or JIS X0201-1976 Roman
1	ESC & @ ESC \$ ) B	JIS X0208-1990 Kanji
2	ESC * I	JIS X0201-1976 Katakana
3	ESC \$ + D	JIS X0212-1990 Supplemental Kanji

Table 5.4: Japanese EUC mapping

([15], [16])

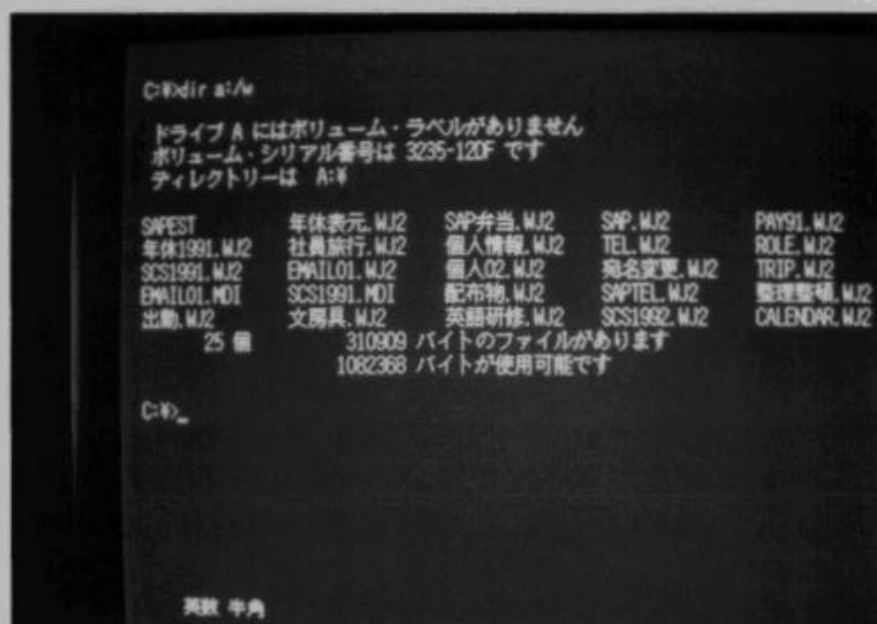
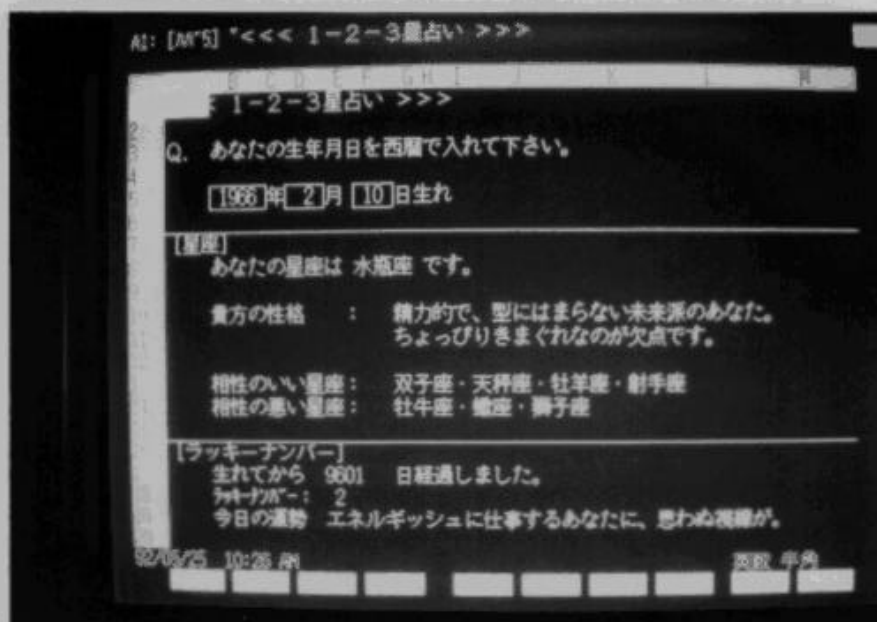


Figure 5.4: Japanese Lotus 123 and Filenames



## 5.3 Ways of Japanization

The way of japanizing a product is always depending on the type of computer system or software you want to "japanize". On the following pages you will find a kind of japanization pathway.

Only some program's (in an English or foreign language version) will be a good seller in Japan without adapting to the local environment. In order to increase the acceptance and sales of the software product you have to do at least some adaptation to the Japanese environment. There are several steps to a fully japanized software package, like :

- A Japanese translation of the manuals, done by a Japanese native speaking technical translator. Sometimes it is helpful to let someone else retranslate the documentation to check the translation of the first translator (see [20]). Besides that it is useful to check the technical terms which are used on the specific platform (e.g., NEC, Toshiba, Fujitsu). This could be considered as the first step to launch a product in Japan. Some special technical applications, which use well known English denominations, do not require more japanization to sell on the Japanese market.
- Enabling the software to work with Japanese characters (Hiragana, Katakana and Kanji) will definitely increase the acceptance of a foreign software product in Japan. For the most program's it is essential that they are enabled to work, display and accept input of Japanese characters. It is possible to write Japanese words in roman characters (Romaji) but the Japanese user will really not be satisfied (similar, for example, to Germany : you can write German umlauts as ae, ue, oe but a German user will not be satisfied to read his name in a different spelling).
- Additional to enabling the software to work with Japanese characters you also should change the program menus and on-line helpscreens. This is a main boost to the software sales in Japan. A Japanese user will be pleased to see a fully japanized foreign software package so that he will consider to buy it. If he has to struggle with a foreign software package, which is not in his own language

he will rather go for a Japanese software package even if it is not so highly developed or sophisticated as the foreign product.

- The highest stage of japanization is to include all the small cultural differences (e.g., date convention, vertical writing, ...). This will show the Japanese user a high commitment of the software developer. This could be the small differences which could influence the Japanese buyer to buy a foreign software package instead of a Japanese program.

Not only program modifications or translations of the manuals are required for a successful competition in the Japanese software market, but also there are some other things which are also quite important :

- Bug free as possible is a major requirement. The Japanese user is not willing to accept major bugs in a program. American or European users are willing to accept bugs because error free is considered as ideal state which is usually not expected and these users are able to cope with that.
- Full, in the most cases for the warranty time free, support is expected from Japanese users. Hotlines, bulletin boards or even salesmen who visit their clients are some ways of support in Japan. If a customer calls your office a salesman or support staff should help him or call back and help to solve the problem.
- Customer training is also expected in Japan. If it is not provided in the form of seminars or classes, it should be provided as computer based training, video or laserdisc course. Sometimes written tutorials are provided with the manuals.
- Some users expect customization or advice for the development of customized applications from the software vendor.

If you show a high commitment to your Japanese customers and provide a good support for the program and for special Japanese features you truly will have a good chance to succeed in the Japanese market place. It is not easy to adapt a program to all the necessary features but the reward will be high.

# Chapter 6

## Further Outlook

As alluded before it is necessary to move towards a common localization standard. At the moment the emerging open systems technology spreads across all types of computer systems. In order to get this technology working across different national computer environments there should be a common standard for, e.g., character sets, locales and EDI (electronic data interchange). A development in this direction is the Unicode (ISO 10646) and the, not so common, TAD (Tron Application Databus character code set).

Other future developments, in Japan, are new input methods like pen input. In addition the development of internationalized software or computer environments will become more and more important. First I want to talk about the new character set proposals for world wide use.

### 6.1 Unicode & TAD

When the US computer hardware and software vendors discovered that about fifty percent of their revenue came from outside the USA they started thinking about a world wide character set standard. This is understandable because a world wide standard would satisfy all requirements of the different national language environments. In addition it would make the software development and localization for different languages much easier. One of the proposals is the ISO 10646 or, so called, Unicode ([18]).

The Unicode was developed mainly by US vendors. Nevertheless contributions came from all over the world. The vendors like Microsoft, SUN and Apple showed a great commitment to the development of this standard. The new versions of their operating systems will probably support this code. Unicode works with a BMP (Basic Multilingual Plane). Each plane has  $256 \times 256$  character positions. There are two proposals for this character set. The UCS (Universal Coded character Set) 2-byte and the UCS 4-byte structure. The UCS-2 supports up to 65536 character positions and the UCS-4 supports a maximum of 2,147,483,647 character positions. Unicode provides the user with control characters, several different national characters, symbols, the, so called, unified CJK (Chinese Japanese Korean) ideographs and an area for private use.

The advantage of the Unicode is that it supports all alphabetical languages on the world and Kanji characters (see figure 6.1 on page 186). The, maybe, disadvantage is that the Chinese, Korean and Japanese Kanji character sets were moved together (unified). The ideas of collecting all Kanji characters from these languages and delete the characters which appear twice or more often sounds good, but you have to consider the different cultural backgrounds. A Kanji which has the meaning A in Japanese could have a totally different meaning in Chinese or Korean. A lot of Asian people are not happy with this approach.

Furthermore the UNIX Systems Laboratory sees, at the moment, no need to support this character standard. It is understandable because they spent a lot of time and money to develop the EUC and the supporting routines. Nevertheless it looks like that the Unicode is one of the best guesses of a new world wide character code set standard.

An other proposed character set is the TAD (Tron Application Databus character code set, see figure 6.2 on page 187). This character set is a proposal from the Tron laboratory of Ken Sakamura (Tokyo University). In this two byte matrix code set it will be possible to handle multiple languages simultaneously ([21]). The TAD character code set is a part of the Japanese Tron project which aims towards a total computer architecture. The Tron project will define, e.g., an international standard for formatting documents, data exchange, data-types and an international character code set (the TAD Multi-lingual character code set). It will support special control codes, different writing directions and a general framework for switching between this language specific environments. In addition it will provide a general framework for

character input.

At the moment it supports the English and the Japanese language. Unfortunately is Tron a development which takes mainly place in Japan. This causes that the Tron project is not well known outside Japans. In addition the most foreign vendors do not support any of the Tron features at the moment. Some foreign companies are observing the activity of the Tron research group, but at the moment the strongest support comes from Japanese companies.

## 6.2 Pen Computing

The difficulties of entering Kanji characters are only partly solved by the use of FEPs. In the office world a lot of the communication is still done by handwriting. The main problem is that it is not very comfortable to enter Kanji characters via a keyboard. The development in Japan goes towards better input methods.

One of the new input methods, with the most impact, is the pen input method. It would be like writing on paper and this would make it more comfortable for the Japanese to enter their Kanji characters. The Kanji characters are always drawn by a fixed stroke order. This makes easier for a pen input recognizing program to detect which character is meant. The development started in Japan about 1970 and today it looks like that pen input is the alternative input method of the future.

## 6.3 Asianization / Internationalization

In a fast moving and changing market like the computer business it will no longer be enough to be present in your home country. The internationalization or globalization of computer systems will open the world wide computer market.

Use this change or lose it. If you not start to work out a concept for the development of global software you will be soon hit by the high costs for adapting your software to different national language environments. In the meantime fifty percent of the revenues from US vendors come from abroad. This is a good sign that the new market for software is global.

### 6.3.1 Asianization

As you have seen in the market section the Japanese market is the biggest market in the Asia/Pacific basin. If you have successfully japanized a product you could use Japan as a base to conquer the rest of the Asian computer market.

Despite the fact that there are differences in the computer systems of Japan, Korea, China and Taiwan they have a lot of things in common. All of them must use a DBCS character set, because all of them rely on Kanji characters. There are different input methods, but in a system like the globalized UNIX versions, your system has not to take care of that. A tool like NLIO supports all of this different country specific specialities.

### 6.3.2 Internationalization / Globalization

If you work on software you should always consider to use the globalized approach. It does not take a long time to develop a product for the world market if you start using localization features from the beginning. If you later on want or have to localize / globalize your product it will be much more expensive and maybe impossible.

The one source concept, which is possible through the use of NLS and NLIO, makes it easy to do this from the beginning. It does not cost much but the reward and the money, which you will save in the future, is worth to do it.

In the PC/WKS market it is visible that both standard operating systems will support the development of internationalized software in the near future. UNIX is already prepared to support localization. The newer versions of MS-Windows show a strong move towards this direction.

In the near future software will be a global business and no longer restricted to one country or area. It is absolutely necessary to change the mind of software designers and programmers to the global approach for software development.

Row-Octet	ISO-646 IRV		Latin-1 Supplement	
00				
01	Extended Latin-A		Extended Latin-B	
02	Extended Latin-B	IPA Extensions	Spacing Modifier Letters	
03	Combining Diacritical Marks		Greek	
04	Cyrillic			
05	Armenian		Hebrew	
06	Arabic			
09	Devanagari		Bengali	
0A	Gurmukhi		Gujarati	
0B	Oriya		Tamil	
0C	Telugu		Kannada	
0D	Malayalam			
0E	Thai		Lao	
10	Tibetan		Georgian	
1E	Additional Extended Latin			
1F	Greek Extensions			
20	General Punctuation	Super-/Subscripts	Currency Symbols	Comb. Diacritical Marks for Symbols
21	Letterlike Symbols	Number Forms	Arrows	
22	Mathematical Operators			
23	Miscellaneous Technical			
24	Control Pictures	O.C.R.	Enclosed Alphanumerics	
25	Box Drawing	Block Elements	Geometric Shapes	
26	Miscellaneous Dingbats			
27	Dingbats			
30	CJK Symbols And Punctuation		Hiragana	Katakana
31	Ecomelo	Hangul Jamo	CJK Miscellaneous	Combining Hangul Jamo
32	Enclosed CJK Letters and Months			
33	CJK Compatibility Words and Hours		CJK Compatibility Abbreviations and Days	
34	Hangul			
3D	Supplementary Hangul			
3E				
45	Old Hangul			
46				
4D	CJK Unified Ideographs			
4E				
9F				
A0				
DF				
E0	Private Use Area			
F7	CJK Compatibility Ideographs			
F9				
FA	Alphabetic Presentation Forms			
FB	Arabic Presentation Forms-A			
FC				
FD				
FE	CJK Compatibility Forms	Small Form Variants	Arabic Presentation Forms-B	
FF	Halfwidth And Fullwidth Forms			Specials

Figure 6.1: Unicode BMP

First Byte	00		21		7E		80		A0		FD		Second Byte
	C0		Reserved						Reserved				
21	Not used		Character Code Zone A  (JIS X0208)				Character Code Zone C						
7E													
80	Not used		Character Code Zone B				Character Code Zone D						
FD													

BTRON1 Japanese language Code Table

Figure 6.2:



# Bibliography

- [1] National Language Support Reference Manual, Volume 2, IBM, SE09-8002-01, USA, 1988
- [2] Hercules /1, Guide to Internationalization, Field Test 2nd Draft, DEC, Maynard, MA, USA, 1991
- [3] Branchenbild, Japan : Der Markt für Minicomputer, Oktober 1986, Köln, Bundesstelle für Aussenhandelsinformation, Nr.10.314.86.442
- [4] Kurzmerkblatt Japan, September 1991, Köln, Bundesstelle für Aussenhandelsinformation, Nr. 0032
- [5] Internationalization in OSF/1 R1.1, Martin, Sandra & Mori, Mariko, OSF, Paper from July 1992, Japan UNIX Symposium
- [6] Large Character Set for C, P.J.Plauger, Article from Dr. Dobbs Journal, August 1992, page 16-24
- [7] Character Code for Japanese Text Processing, Journal of Information Processing, Information Processing Society of Japan, Vol.13, No.1, 1990, Paper by Akira Miyazawa
- [8] Japan.inf version 1.2 by Ken Lunde Adobe Systems Intl., March 1992
- [9] National Character Processing, Tutorial, Nihon Unisys, 1992
- [10] Softlanding in Japan, Version 1.0J, American Electronics Association, 1991, Tokyo
- [11] Softlanding in Japan, Version 2.0J American Electronics Association, 1992, Tokyo

- [12] Global Software by Dave Taylor, Springer, 1992, Menlo Park CA
- [13] UNIX System V Release 4 Multi National Language Supplement, AT&T Unix System Laboratories, Tokyo, 1990
- [14] DYNIX/ptx Native Language Support Programming Guide, Sequent Computer Systems, Order No. 1003-50889-00
- [15] Definition of Japanese EUC, UI-OSF-USL Joint Announcement, Press Release, 12/12/1991
- [16] Internationalization and Localization on UNIX System V, Unix System Laboratories Pacific, Jiro Monden, 1/2/1992 (Hand-outs)
- [17] HP-UX NLS A Guide to Internationalization of Software, Hewlett Packard, Japan, 1989
- [18] ISO/IEC 2nd DIS 10646 Universal Coded Character Set, Slides from Nihon DEC, Tokyo, Japan, 1992
- [19] IDC Report Japan, IDC, Tokyo, Japan, 1988-92
- [20] Porting PC Software for the Japanese Market, Phil Grouse, Stylus Software, Australian Computer Conference 1988: Information Technology Towards 2000. Proceedings, p.263-70, Australian Comput. Soc, Darlinghurst, NSW, Australia
- [21] The Tron Multilingual Computing Project, University of Tokyo, Sakamura Laboratory
- [22] NEC PC-9800 Quick Guide, PC-98 Developer Support Program USA, XLsoft Intl., Irvine CA, 1992
- [23] Information from NEC Tokyo PC-98 Developer Support Program
- [24] Addendum to Technical Guide to Asian Language Software Localization, DEC, Hong Kong, 1990
- [25] JIS X0201 Code for Information Interchange, JSA, Tokyo, Japan 1976
- [26] JIS X0208 Code of the Japanese Graphic Character Set for Information Interchange, JSA, Tokyo, Japan, 1983

- [27] JIS X0212 Code of the supplementary Japanese graphic character set for information interchange, JSA, Tokyo, Japan, 1990
- [28] Keyboard Layout for Information Processing JIS X6002 & JIS X6004, JSA, Tokyo, Japan 1980, 1986
- [29] JIS X0301 Identification Code of Dates, JSA, Tokyo, Japan, 1977
- [30] JIS X0302 Identification Code of Times, JSA, Tokyo, Japan, 1977
- [31] Migrating VMS applications to the Japanese VMS environment, Burkley, R.E.,: DEC, Acton, MA, USA, Electro/88 Conference Record p.13/1/1-8,
- [32] IDC world wide survey of the IT market, IDC, 1991
- [33] Japan Computer Quarterly No. 76, JIPDEC, Tokyo, Japan, 1989
- [34] Japan's Personal Computer Software market, JPL, Tokyo, Japan, 1991
- [35] Jetro Your Martket in Japan No. 79
- [36] Information Service Industry in Japan, JISA, Tokyo, Japan 1991
- [37] Computer Service Industrie, British Embassy, 1990
- [38] Informatization White Paper, Softic, Tokyo, Japan, 1991

# Appendix A

## Index

# Index

50-On Sequence, 142

abstarct, 0

Address format, 136

Amikake, 134

ANSI C, 170

ASCII, 60

Asianization, 184

Aufgabenstellung, 1

AX, 167

BMP, 183

Business etiquette, 53

Byte oriented, 88

Calendar, 106

Character oriented, 88

Cho, 101

Clean programs, 163, 167

Collating Sequence, 142

Consensus, 54

Conversion, 115

CSP, 169

Currency, 101

Custom build Software, 55

Customized Software, 19

Dakuon, 134

DBCS, 62

DBCS string handling, 86

Denotation, 145

Dirty programs, 163

DOS/V, 167

emperor, 105

En, 102

ESC/P, 139

EUC, 169, 171

EUC representation, 171

FEP, 115

Front End Processor, 115

Furigana, 134

Gaji, 64

Gengou, 105

Globalization, 185

Gogo, 108

Gozen, 108

Gregorgian calendar, 104

Handakuon, 134

Hankaku, 60, 133

Hardware, 138

Hepburn, 58

Hiragana, 58

HP-UX, 172

Hyaku, 101

I-RO-HA, 142

IBM PC, 17, 141

- Internationalization, 185
- Japanese business style, 53
- Japanese Data-type, 143
- Japanese features, 164
- Japanese Hardware, 16
- Japanese Input Manager, 119
- Japanese language, 12
- Japanese language problems, 56
- Japanese market, 14
- Japanese software style, 19
- Japanese writing style, 130
- Japanization, 163
- japanization, 12
- Japanization requirements, 56
- JIM, 119
- JIS X 0208, 143
- JIS X 0212, 143
- JIS X0201, 59
- JIS X0208, 60
- JIS X0212, 61
- Joyo Kanji table, 61
- Ju, 101
- Jukugo Kanji conversion, 118
- Kanji, 58
- Kanji dictionary, 118
- Kanji In / Out, 62
- Kansuji, 101, 102, 133
- Katakana, 58
- Keisen, 133
- keywords, 0
- Kinsoku Shori, 133
- Kinto-waritsuke, 135
- Kun-yomi, 142
- LANG, 170
- Laptop, 17
- Locale, 170
- Localization features, 170
- Lotus 123j, 145, 167
- Man, 101
- Meishi, 54
- Message catalog, 173
- Mixed string, 86
- MNLS, 169
- MS-DOS, 166
- MS-Windows, 168
- Multi-byte, 172
- National profile, 170
- NEC PC-98XX, 17, 141
- Nengou, 105
- Nipponsiki, 58
- NLIO, 174
- NLS, 172
- Notebook, 17
- Number, 101
- OADG, 167
- Oku, 101
- On-yomi, 142
- Package Software, 20
- Page order, 130
- Paper size, 138
- PC/WKS, 14
- PC/WKS market, 16
- Pen computing, 184
- Pitfalls, 144
- Poem, 130

- Quality, 55
- Radical, 143
- Reign, 105
- Renbusetsu Kanji conversion, 118
- Romaji, 58
- Rubi, 134
- SBCS, 62
- SBCS string handling, 86
- Sen, 101, 102
- Service, 55
- Shift In / Out, 62, 87
- Shift JIS, 63, 171
- Single Shift code, 171
- Small Kanas, 134
- Sokuon, 134
- Sort order, 142
- TAD, 182
- Tan Kanji conversion, 117
- Tenno, 105
- Time, 107
- topics covered, 10
- Tron, 182
- Unicode, 182
- Units, 137
- UNIX, 166, 168, 183
- Western writing style, 130
- Wide-character, 171
- World IT market, 15
- Yen, 102
- Yo'on, 134
- Yomi, 58
- Zenkaku, 133

# Appendix B

## Glossary

**ALE** Asian Language Environment

**Amikake** is the gray-shading of characters, used instead of boldface, italics or underlining.

**ANSI** (American National Standard Institute) is the standardization organization of the USA

**ASCII** (American Standard Code for Information Interchange) standard seven bit code character set.

**Back Translation** is to retranslate a document from Japanese to check if the first translation is correct.

**Bushu** is the collating sequence based on the type of strokes which are used to draw the Kanji character.

**Character Set** contains all letters, ideographs, numbers, punctuation symbols, special symbols, etc. which are required to write a language.

**Codeset** is a set of characters which are match to a unique, uniform length binary code.

**Country Specific Parts** is a term used in the AT&T Unix environment and describes the parts of the **MNLS** which are unique for a country, like the **FEP** for Japan.



**CSP** see Country Specific Parts

**DBCS** (Double Byte Character Set) is a set of characters (**codeset**) which is represented by two Bytes (16 Bit).

**DIN** (Deutsche Industrie Norm) is the standardization organization of Germany

**EUC** (Extended Unix Code) a way of mapping different character sets. Allows the use of up to four bytes for the representation of one character. Organized in four code sets. Code set number 0 is for default ASCII.

**En** see **Yen**

**External Codes** are the representation which is used for the input and output of **EUC** mapped character sets.

**FEP** see **Front End Processor**

**Front End Processor** is a program which accepts the user input and react in a certain form. Depending on the mode it passes the character through or converts the entered characters to Hiragana, Katakana or via the Kana Kanji dictionary to Kanji.

**Furigana** see **Rubi**

**Gaiji** are from the user self defined (Kanji) ideographic characters. It is used to display or print non standard (Kanji) characters which are not included in the Japanese Character set standard.

**Gairaigo** see **Loan-word**

**Gengou** see **Nengou**

**Globalization** approach to design software that this software is able to work worldwide in different cultural, language environments.

**Gojuuonjun** (50 on table) is a way of sorting in Japan. This table is used to sort after the pronunciation of the Kanji.

**Hankaku** is mainly used for **Katakana** characters. It is a way to print **Katakana** in the same size as **ASCII** characters instead of printing them in **Zenkaku**.

**Hepburn** way of transliterating between Japanese words and their written representation in **romaji**.

**Hiragana** is a syllable alphabet which is used in Japan to write Japanese Terms and to add grammatical constructions to Kanji characters. Hiragana contains about 83 characters.

**I18N** is the short-cut for **Internationalization**

**IBMSCII** is character set of the IBM PC. Contains, beside the standard ASCII characters, some European character extensions, line elements and special characters.

**Ideograph** is a character which represents a picture, thought or meaning by itself. This ideograph usually does not include any information about the pronunciation or name of the thing described.

**Internal Codes** is the way of storing **EUC** mapped character's sets in memory.

**Internationalization** is to design software which can be adapted to different language environments without major modification.

**I-RO-HA** is a collating (sort) sequence used in Japan.

**ISO** (International Standardization Organization) is an international standardization organization formed by standardization organization (ANSI, JIS, DIN, BSI, ...) from different countries.

**ISO 10646** see **Unicode**

**JAE** see **Japanese Application Environment**

**Japanese Application Environment** is a term used to describe a UNIX environment which works with **EUC** mapped character sets.

**Japanese Input Manager** is the combination of **FEP** and Kana Kanji dictionary, realized in different ways (e.g., as a part of the keyboard driver or as a separate process)

**JIM** see Japanese Input Manager

**JIS** (Japan Industrie Standard) is the Japanese body for standardization.

**JIS Code** is a industrie standard **codeset** defined from **JIS**. Some examples are JIS X0208 or JIS X0212. JIS X0208 Level 1 contains 2.965 Kanji characters and Level 2 contains 3.388 Kanji character's.

**Jukugo (Kanji Compound) Conversion** is the way of entering Kanji character's compounds by their Kana spelling. So that it is possible to enter Kanji compounds which consist more than one Kanji character.

**Kana-Kanji Conversion** is the conversation from **Kana** (or **Romaji**) alphabet to Kanji ideographs. The input is converted via a dictionary to a Kanji which has the pronunciation of the input.

**Kana** is a short cut for the **Hiragana** and **Katakana** syllable character sets.

**Kango** are words derived from Chinese.

**Kanji** characters are **ideographs**. A Kanji represents a meaning, idea or thought but not the pronunciation.

**Kanji Dictionary** is the dictionary which is used to convert the Kana or Romaji input of the user to Kanji characters.

**Katakana** is a syllable alphabet which is used in Japan to write foreign words, denominations or loan-words. Katakana contains 86 characters.

**Keisen** are lines, boxes or borders which the Japanese draw around table, lists and reports.

**Kinsoku** is a type of word wrapping which prevents that certain characters appear at the beginning or end of a printed line (line head and line end kinsoku processing).

**Kintou-Waritsuke** is a form of formatting Japanese text between two points, e.g., the begin and end of a line.

**Kun Reading** is the way of the Japanese pronunciation of a Kanji.

**Kunrei-Shiki** see **Hebrun**, different method

**Kuten Code** is a code which is used to enter Japanese Kanji characters. Usually the Kanji characters are listed in a table with, e.g., their Kuten code.

**LANG** is the UNIX language environment variable which contains the information in which language environment a program is running. It is used to determine which cultural specific differences have to be considered.

**Loan-word** is a word from a foreign language which is adapted and frequently used in Japan. An example is software which has become the loan-word, e.g., *sofuto uetā*

**Locale** is a set of locale definitions which are apply for a locale environment. Usually the locale in a UNIX system is selected by an environment variable called LANG.

**Localization** is the process of adapting a software product to a national, ethnic, language environment. It could contain changes of date & time formats, . . . .

**MNLS** (Multi National Language Support) is way which vendors have chosen to support different national, cultural and language environments.

**Multibyte Character** is a way of representing characters with more then one byte. An example for multi-byte character sets are **EUC** or JIS X0212.

**Nengou** is the Japanese calendar which is based on the reign of the emperor (tenno).

**Nippon-siki** see **Hepburn**, different method

**NLS** see **MNLS**

**On Reading** is the way of the Chinese to pronounce a Kanji character.

**Radical** is a set of 214 basic Kanji characters which are used to build more complex characters.

**Regionalization** is to adapt a software to special requirements of a region (like enabling DBCS in Asia)

**Renbunsetsu (Phrase) Conversion** is to convert the input of whole phrases (containing Kanji and Kanas) in one step.

**Romaji** is the Japanese name for the standard western alphabet. In the JIS standard code-set it is equal to the **ASCII** codeset.

**Rubi** are little characters which are used to give the reader the pronunciation of a not so frequently used **Kanji**.

**SBCS** (Single Byte Character Set) is a codeset which is based on one byte, e.g., **ASCII** or **IBMSCII**.

**Shift JIS** is a derivation of the standard JIS codeset. The positions of the characters are moved to an other location in the two byte matrices.

**Sokaku** is the sorting order based on the numbers of strokes which are used to draw the Kanji character.

**Tan Kanji (Single Kanji) Conversion** is an old method of entering Kanji characters. After entering the Kana spelling for one Kanji this spelling will be converted, so that it is possible to enter Kanji character's one by one.

**To Reading ???**

**Unicode** is a proposal for a codeset standard which contains all character's/code sets which are needed worldwide (is equal to ISO10646).

**Vertical Writing** is a heavily used writing system in Japan. The text is written from the right side of a page to the left side and from the top of a page to their bottom.

**Wago** are original (native) Japanese terms which where in use before the introduction of **Kanji** characters from China.

**Wide Character** is an, in **ANSI C** declared data-type for the use with **Multibyte** character code sets.

**YEN** is the name of the Japanese currency (in Japan also called EN). 1 YEN is 100 SEN but this is mainly used for banking purposes.

**Yomi** is the pronunciation of Japanese Kanji characters and words.

**Zenkaku** characters have the doubled width of, e.g., **ASCII** characters. All Hiragana and Kanji are printed in the Zenkaku size.

# Appendix C

## Character Sets

On the next couple of pages you will find some examples for the coding of JIS code, EUC and Shift JIS. The examples show only a short part of these character sets. The first row gives us the character code in Hex.

21 20! 、 。 , . . : ; ? ! \* ^ ` ~`  
21 30! ^ \_ ` ˇ ˘ ˙ ˚ 全々 / ○ — — /  
21 40! \ ~ || | …… ‘ ’ “ ” ( ) [ ] []  
21 50! { } < > 〈〉 「 」 『 』 【 】 + - ± ×  
21 60! ÷ = ≠ < > ≤ ≥ ∞ ∴ ♂ ♀ ° ´ ¨ ℃ ¥  
21 70! \$ ¢ £ % # & \* @ § ☆ ★ ○ ● ◎ ◇  
22 20! ◆ □ ■ △ ▲ ▽ ▼ ※ 〒 → ← ↑ ↓ =  
22 30! ∈ ∋ ⊆ ⊇ ⊂ ⊃  
22 40! ∪ ∩ ∧ ∨ ¬ ⇒ ⇔ ∇  
22 50! ∃ ∠ ⊥ ∩ ∩  
22 60! ∇ ≡ ≐ ≪ ≫ √ ∞ ∝ ∴ ∫ ∫  
22 70! Å % # b ♭ † ‡ ¶ ○  
23 20!  
23 30! 0 1 2 3 4 5 6 7 8 9  
23 40! A B C D E F G H I J K L M N O  
23 50! P Q R S T U V W X Y Z  
23 60! a b c d e f g h i j k l m n o  
23 70! p q r s t u v w x y z  
24 20! ああいううええおおかがきぎく  
24 30! ぐけげこごさざしじすずせぜそぞた  
24 40! だちぢっつづてでとどなにぬねのは  
24 50! ばばひびびふぶおへべほほまみ  
24 60! むめもややゆゆうよよりるれろわわ  
24 70! ゐゑをん  
25 20! アアイイウウエエオオカガキギク  
25 30! グケゲコゴサザシジスズセゼソゾタ  
25 40! ダチヂッツツヅテデトドナニヌネノハ  
25 50! ババヒビピフブプヘベペホボポマミ  
25 60! ムメモヤユユヨヨラリルレロワッ  
25 70! キエランヴカケ  
26 20! Α Β Γ Δ Ε Ζ Η Θ Ι Κ Λ Μ Ν Ξ Ο  
26 30! Π Ρ Σ Τ Υ Φ Χ Ψ Ω  
26 40! α β γ δ ε ζ η θ ι κ λ μ ν ξ ο  
26 50! π ρ σ τ υ φ χ ψ ω  
26 60!  
26 70!

Figure C.1:

27 20: А Б В Г Д Е Ё Ж З И Й К Л М Н  
 27 30: О П Р С Т У Ф Х Ц Ч Ш Щ Ъ Ы Ь Э  
 27 40: Ю Я  
 27 50: а б в г д е ё ж з и й к л м н  
 27 60: о п р с т у ф х ц ч ш щ ъ ы ь э  
 27 70: ю я  
 28 20: — | Г Г Л Т Т Т Т Т Т Т Т Т  
 28 30: Л Т Т Т Т Т Т Т Т Т Т Т Т Т Т  
 28 40: 十  
 30 20: 垂啞娃阿哀愛挨始逢葵茜穉惡握渥  
 30 30: 旭葦芦蓼梓庠幹扱宛姐虻飴綯綾鮎或  
 30 40: 粟裕安庵按暗案闇鞍杏以伊位依偉囿  
 30 50: 夷委威尉惟意慰易椅為畏異移維緯胃  
 30 60: 萎衣謂違遺医井亥域育郁磯一壹溢逸  
 30 70: 稻茨芋鰯允印咽員因姻引飲淫胤蔭  
 31 20: 院陰隱韻吋右宇烏羽迂雨卯鵜窺丑  
 31 30: 碓臼渦噓唄鬱蔚鰻姥厖浦瓜閨噂云運  
 31 40: 雲荏餌穀營嬰影映曳榮永泳洩瑛盈穎  
 31 50: 穎英衛詠銳液疫益馭悅謁越閱榎厭円  
 31 60: 園堰奄宴延怨掩援沿演炎焰煙燕猿緣  
 31 70: 艷苑菌遠鉛鴛塩於汚甥凹央奧往応  
 32 20: 押旺橫欧殴王翁襖鶯鷗黃岡冲荻億  
 32 30: 屋憶臆桶牡乙俺卸恩溫穩音下化佻何  
 32 40: 伽伽佳加可嘉夏嫁家寡科暇果架歌河  
 32 50: 火珂禍禾稼箇花苛茄荷華菓蝦課嘩貨  
 32 60: 迦過霞蚊俄峨我牙画臥芽蛾賀雅餓駕  
 32 70: 介会解回塊壞迴快怪悔恢懷戒拐改  
 33 20: 魁晦械海灰界皆繪芥蟹開階貝凱劾  
 33 30: 外咳害崖慨概涯碍蓋街該鎧骸湮馨蛙  
 33 40: 垣柿蛎鈎劃嚇各廓扞攪格核殼獲確穫  
 33 50: 覺角赫較郭閣隔革学岳樂額顎掛笠桎  
 33 60: 櫃梃鯁渴割喝恰括活渴滑葛褐轄且鯉  
 33 70: 叶枇樺鞞株兜竈蒲釜鎌囓鴨栢茅萱  
 34 20: 粥刈苕瓦乾侃冠寒刊勘勸卷喚堪姦  
 34 30: 完官寬干幹患感慣憾換敢柑桓棺款歛  
 34 40: 汗漢澗灌環甘監看竿管簡緩缶翰肝艦  
 34 50: 莞觀諫貫還鑑間閑閑陷韓館館丸含岸  
 34 60: 巖玩癌眼岩翫贗雁頑顏願企伎危喜器  
 34 70: 基奇嬉寄岐希幾忌揮机旗既期棋棄

Figure C.2:



EUC-CODIERUNG

a1 a0: ` , . : ; ? ! \* ' ^ ~  
 a1 b0: ^ \_ ` ~ ` ` 全々 / 〇 - - - /  
 a1 c0: \ ~ || | ... ' ' " ( ) [ ]  
 a1 d0: {} < > < > 「 」 『 』 【 】 + - ± ×  
 a1 e0: ÷ = ≠ < > ≤ ≥ ∞ ∴ ♂ ♀ ° ´ ˆ ° ™ ¥  
 a1 f0: \$ ¢ £ % # & \* @ § ☆ ★ ○ ● ◎ ◇  
 a2 a0: ◆ □ ■ ▲ ▴ ▽ ▾ ※ → ← ↑ ↓ =  
 a2 b0: ∈ ∋ ⊆ ⊇ ⊂ ⊃  
 a2 c0: ∪ ∩ ∧ ∨ ¬ ⇒ ⇔ ∇  
 a2 d0: ∃ ∠ ⊥ ∩ ∂  
 a2 e0: ∇ ≡ ≐ ≪ ≫ √ ∞ ∞ ∴ ∫ ∫  
 a2 f0: Å % # b ♪ † ‡ ¶ ○  
 a3 a0:  
 a3 b0: 0 1 2 3 4 5 6 7 8 9  
 a3 c0: A B C D E F G H I J K L M N O  
 a3 d0: P Q R S T U V W X Y Z  
 a3 e0: a b c d e f g h i j k l m n o  
 a3 f0: p q r s t u v w x y z  
 a4 a0: ああいううええおおかがきぎく  
 a4 b0: ぐけげこごさざしじすずせぜそぞた  
 a4 c0: だちちつつづてでとどなにぬねのは  
 a4 d0: ばばひびびふぶおへべへほほほまみ  
 a4 e0: むめもややゆゆよよらりるれろわわ  
 a4 f0: ゐゑをん  
 a5 a0: アアイイウウエエオオカガキギク  
 a5 b0: グケゲコゴサザシジスズセゼソゾタ  
 a5 c0: ダチヂッツツテデトドナニヌネノハ  
 a5 d0: バパヒビピフブフヘベベホボボマミ  
 a5 e0: ムメモヤヤユユヨヨラリルレロワワ  
 a5 f0: キエランヴカケ  
 a6 a0: Α Β Γ Δ Ε Ζ Η Θ Ι Κ Λ Μ Ν Ξ Ο  
 a6 b0: Π Ρ Σ Τ Υ Φ Χ Ψ Ω  
 a6 c0: α β γ δ ε ζ η θ ι κ λ μ ν ξ ο  
 a6 d0: π ρ σ τ υ φ χ ψ ω  
 a6 e0:  
 a6 f0:

Figure C.3:



SHIFT-JIS CODIERUNG

81 3f! ` , . : ; ? ! \* ' ^ ~  
 81 4f! ^ \_ ` ˘ ˇ ˙ ˚ 全々 / ○ — — /  
 81 5f! \ ~ || | ... ' " ( ) [ ] []  
 81 6f! || < > < > 「 」 『 』 【 】 + - ± ×  
 81 80! ÷ = ≠ < > ≤ ≥ ∞ ∴ ♂ ♀ ° ´ ˆ ° ™ ¥  
 81 90! \$ ¢ £ % # & \* @ § ☆ ★ ○ ● ◎ ◇  
 81 9e! ◆ □ ■ ▲ ▼ ※ 〒 → ← ↑ ↓ =  
 81 ae! ∈ ∋ ⊆ ⊇ ⊃ ⊂  
 81 be! ∪ ∩ ∧ ∨ ⇔ ⇔ ⇔ ⇔  
 81 ce! ∃ ∠ ⊥ ^ ∂  
 81 de! ∇ ≡ ≐ < > √ ∞ ∞ ∴ ∫ ∫  
 81 ee! Å % # b ♪ † ‡ ¶ ○  
 82 3f!  
 82 4f! 0 1 2 3 4 5 6 7 8 9  
 82 5f! A B C D E F G H I J K L M N O  
 82 6f! P Q R S T U V W X Y Z  
 82 80! a b c d e f g h i j k l m n o  
 82 90! p q r s t u v w x y z  
 82 9e! ああいうえおおかがきぎく  
 82 ae! ぐけげこごさざしじすずせぜそぞた  
 82 be! だちぢっつづてでとどなにぬねのは  
 82 ce! ばばひびびふぶへべほほまみ  
 82 de! むめもややゆゆよよらりるれろわわ  
 82 ee! ゐゑをん  
 83 3f! アアイイウウエエオオカガキギク  
 83 4f! グケゲコゴサザシジスズセゼソゾタ  
 83 5f! ダチヂッツツテデトドナニヌネノハ  
 83 6f! ババヒビピフブプヘベベホボボマミ  
 83 80! ムメモヤヤユユヨヨラリルレロワワ  
 83 90! キエヲンヅカケ  
 83 9e! Α Β Γ Δ Ε Ζ Η Θ Ι Κ Λ Μ Ν Ξ Ο  
 83 ae! Π Ρ Σ Τ Υ Φ Χ Ψ Ω  
 83 be! α β γ δ ε ζ η θ ι κ λ μ ν ξ ο  
 83 ce! π ρ σ τ υ φ χ ψ ω  
 83 de!  
 83 ee!

Figure C.5:

Figure C.6:

# Appendix D

## Useful Addresses

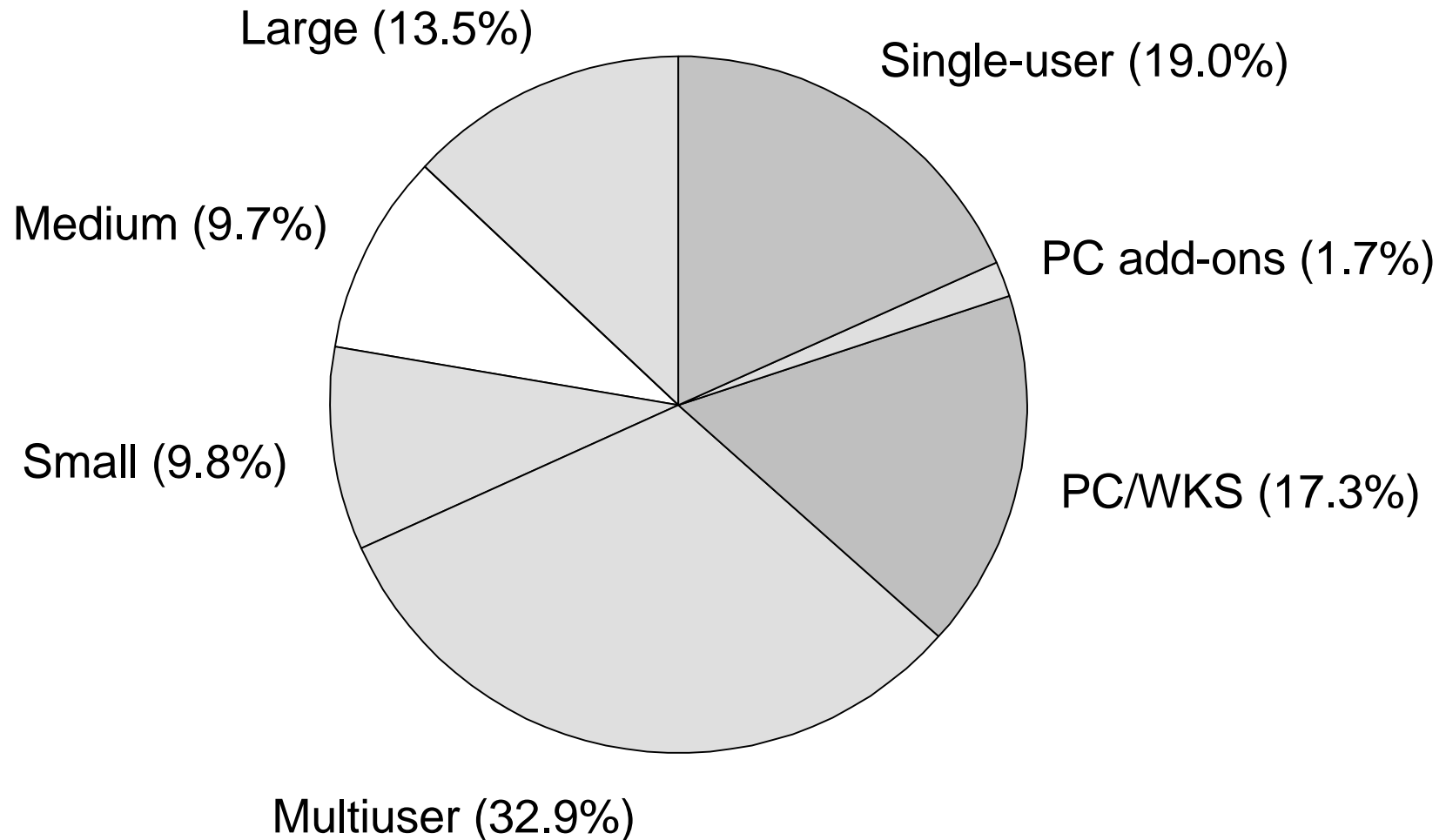
On the next pages you will find some useful addresses which maybe can help you to gather information about the Japanese hard- & software market.

NTT Systems Technologies Inc.	7th Fl. World Import Mart
3rd Fl. Nihonbashi Muromachi Bldg	1-3 Higashi Ikebukuro 3-chome
3-16 Nihonbashi Hongoku-cho 3-chome	Toshima-ku Tokyo $\overline{T}$ 170 Japan
Chuo-ku Tokyo $\overline{T}$ 103 Japan	Software AG of Far East Inc.
Software Consultant Corp. K.K.SCC	Yaesu Mitsui Bldg.
EDC Bldg 62-1 Nakano 5-chome	7-2 Yaesu 2-chome
Nakano-ku Tokyo $\overline{T}$ 164 Japan	Chuo-ku Tokyo $\overline{T}$ 104 Japan
Software Development Corp.Ltd	Japan Information Center of Science and
Shibuya Daiichi Seimei Bldg.	Technology JICST
8-12 Shibuya 3-chome	5-2 Nagata-cho 2-chome
Shibuya-ku Tokyo $\overline{T}$ 150 Japan	Chiyoda-ku Tokyo $\overline{T}$ 100 Japan
Software Management Co.Ltd	NTT Data Communication Systems
23rd Fl. Shinjuku Green Tower Bldg.	Corp.
14-1 Nishi Shinjuku 6-chome	No.17 Mori Bldg.
Shinjuku-ku Tokyo $\overline{T}$ 160 Japan	26-5 Toranomom 1-chome
Software Research Assosciates Inc.	Minato-ku Tokyo $\overline{T}$ 105
1-1 Hirakawa-cho 1-chome	Unix International Inc.
Chiyoda-ku Tokyo $\overline{T}$ 102 Japan	Korufuru Kanda
Microsoft Co.Ltd	2-1 Kanda Suda-cho 1-chome
16th Fl. K Bldg.	Chiyoda-ku Tokyo $\overline{T}$ 101
5-25 Nishi Shinjuku 7-chome	Japanese Standards Assn.
Shinjuku-ku Tokyo $\overline{T}$ 160 Japan	1-24 Akasaka 4-chome
European Business Community	Minato-ku Tokyo $\overline{T}$ 107
New Hoechst Bldg.	JETRO
10-16 Akasaka 8-chome	2-5 Toranomom 2-chome
Minato-ku Tokyo $\overline{T}$ 107 Japan	Minato-ku Tokyo $\overline{T}$ 105
Delegation of the Commission	JAPAN
of the European Communities	JECC
Europa House	Japan Electronic Computer Co.Ltd
9-15 Sanban-cho	Shin-Kokusai Bldg.
Chiyoda-ku Tokyo $\overline{T}$ 102 Japan	3-4-1 Marunouchi Chiyoda-ku
US Export Development Office	Tokyo $\overline{T}$ 100 JAPAN

Ministry of International Trade & Industry	Japan Data Communications Association
Machinery & Information Industries Bureau	25-10 Chuo 4-chome Nakano-ku
1-3-1 Kasumigaseki Chiyoda-ku	Tokyo $\overline{T}$ 164 JAPAN
Tokyo $\overline{T}$ 100 JAPAN	Borland Japan Co.Ltd.
JEIDA	Odakyu Minami Aoyama Bldg. 9th Fl
Japan Electronic Industry Development Association	7-8-1 Minami Aoyama Minato-ku
Kikai Shinko Kaikan	Tokyo $\overline{T}$ 107 JAPAN
3-5-8 Shibakoen Minato-ku Tokyo $\overline{T}$ 105 JAPAN	Science and Technology Agency (STA)
Nippon Computer Graphics Association	2-2-1 Kasumigaseki Chiyoda-ku
1-2-2 Uchikanda Chiyoda-ku	Tokyo $\overline{T}$ 100 JAPAN
Tokyo $\overline{T}$ 101 JAPAN	Nihon Sun Microsystems K.K
JASA	Kowa Nibancho Bldg.
Japan System-house Association	11-19 Niban-cho Chiyoda-ku
18-12 Hakozaiki-cho Nihonbashi Chuo-ku	Tokyo $\overline{T}$ 102 JAPAN
Tokyo $\overline{T}$ 103 JAPAN	ASCII Corp
K.K. Ashisuto	Three F Minami Ayoma Bldg.
(attn: Bill Totten)	11-1 Minami Aoyama 6-chome
1-1 Manpukuji 1-chome	Minato-ku Tokyo $\overline{T}$ 106 JAPAN
Asaoku Kawasaki City 215 JAPAN	Micro Software Associates
Softbank	Oda Kyo Minami Ayoma Bldg.
Ns Takanawa Bldg.	8-1 Minami Ayoma 7-chome
19-13 Takanawa 2-chome Minato-ku	Minato-ku Tokyo $\overline{T}$ 107 JAPAN
Tokyo $\overline{T}$ 108 JAPAN	NSF/NAESIS
Centre of the International Cooperation for Computerization (CICC)	(Attn: Lawrence A. Grafield)
Mita No.43 Mori Bldg 15f	Room 416A NSF
13-16 Mita 3-chome Minato-ku	1800 G Street Washington DC 20550 USA
Tokyo $\overline{T}$ 108 JAPAN	Unix System Laboratories Pacific Ltd.
	No.1 Nan-Oh Bldg.
	21-2 Nishi Shinbashi 2-chome Minato-ku
	Tokyo $\overline{T}$ 105 JAPAN

# Proportions of IT Market 1990

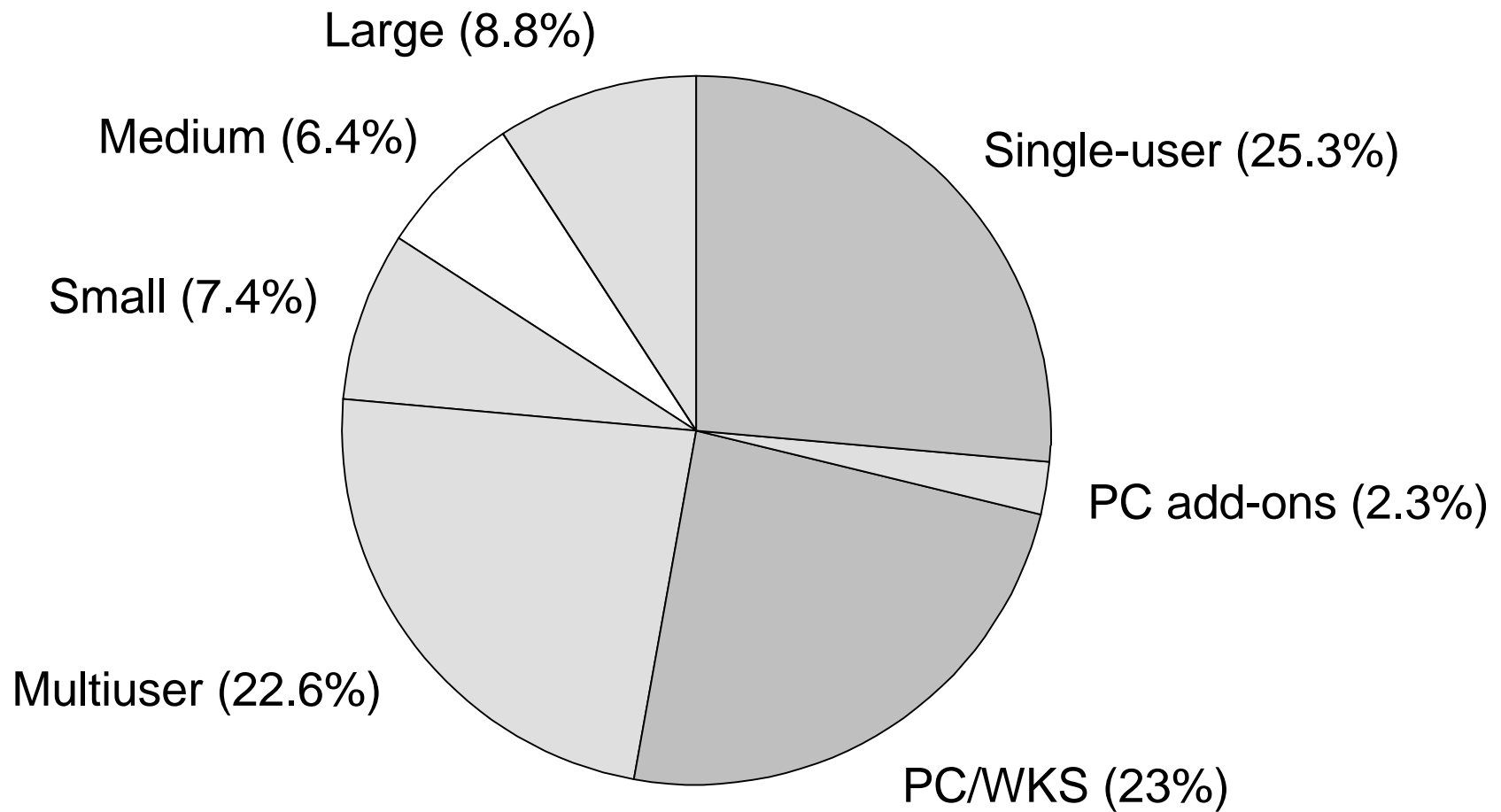
## Hardware in Japan



Source: IDC, 1991

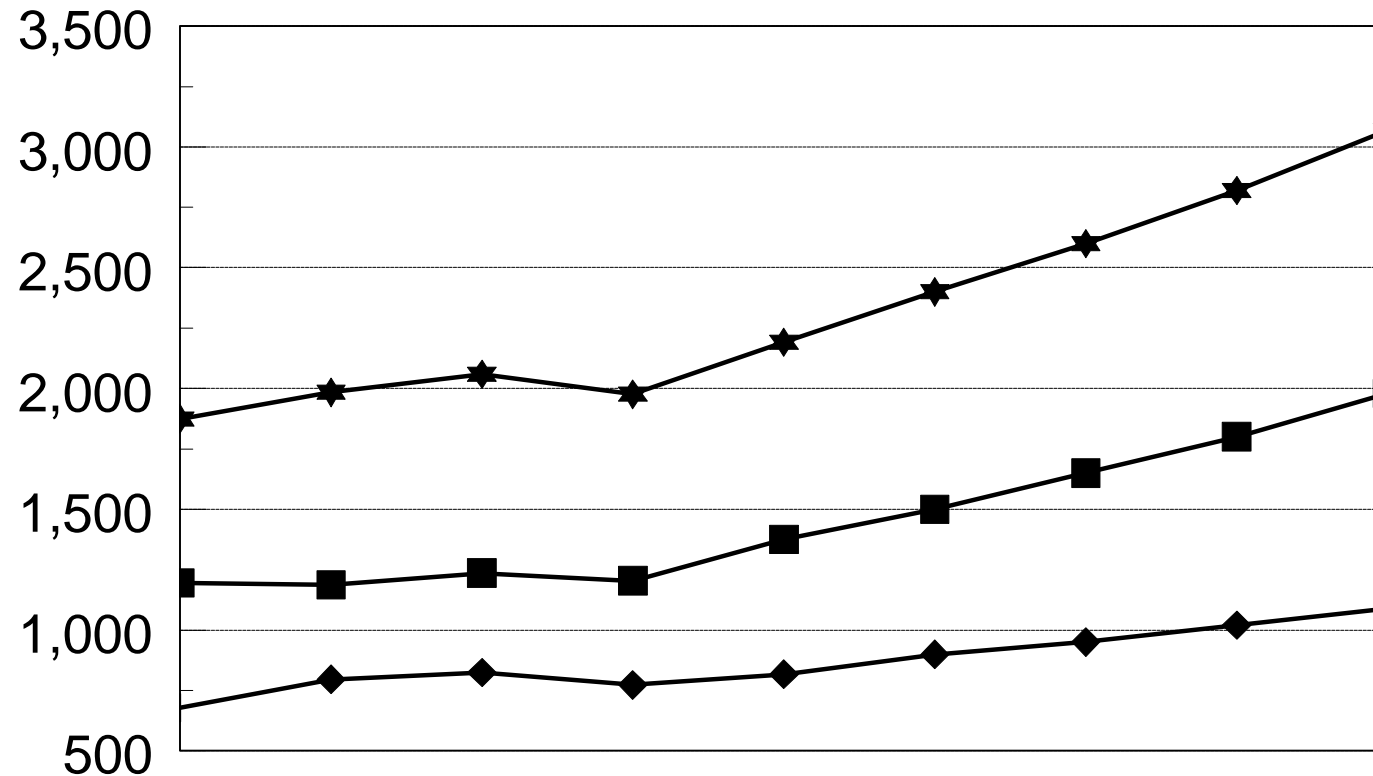


# Proportions of IT Market 1990 Hardware in USA



Source: IDC, 1991

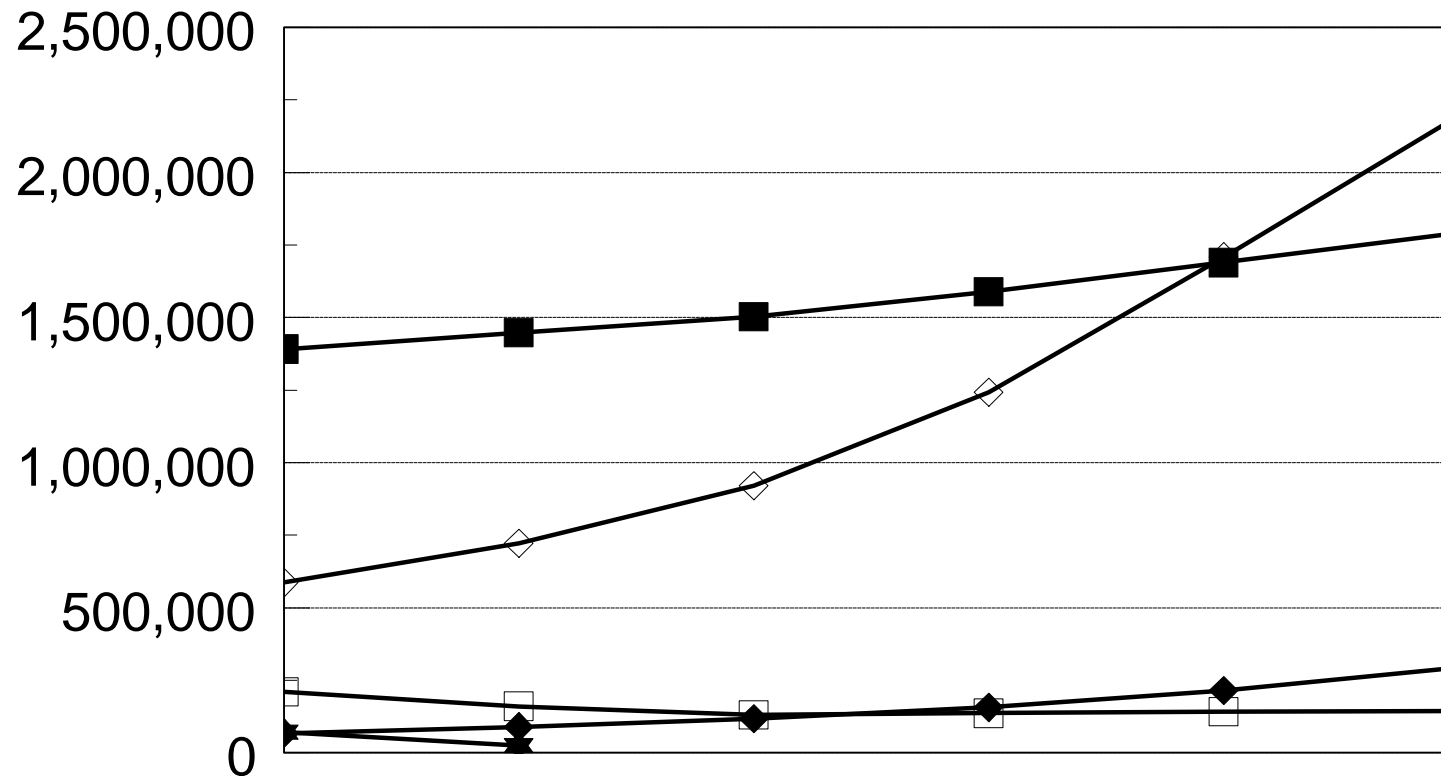
# Total Shipment of PC's in Japan



	1984	1985	1986	1987	1988	1989	1990	1991	1992
Domestic ■	1,196	1,187	1,236	1,204	1,375	1,500	1,650	1,800	1,980
Export +	678	796	824	773	817	900	950	1,020	1,090
Total ★	1,874	1,984	2,059	1,976	2,191	2,400	2,600	2,820	3,070

Sources: JISA, Units : 1000 PC's, estimated

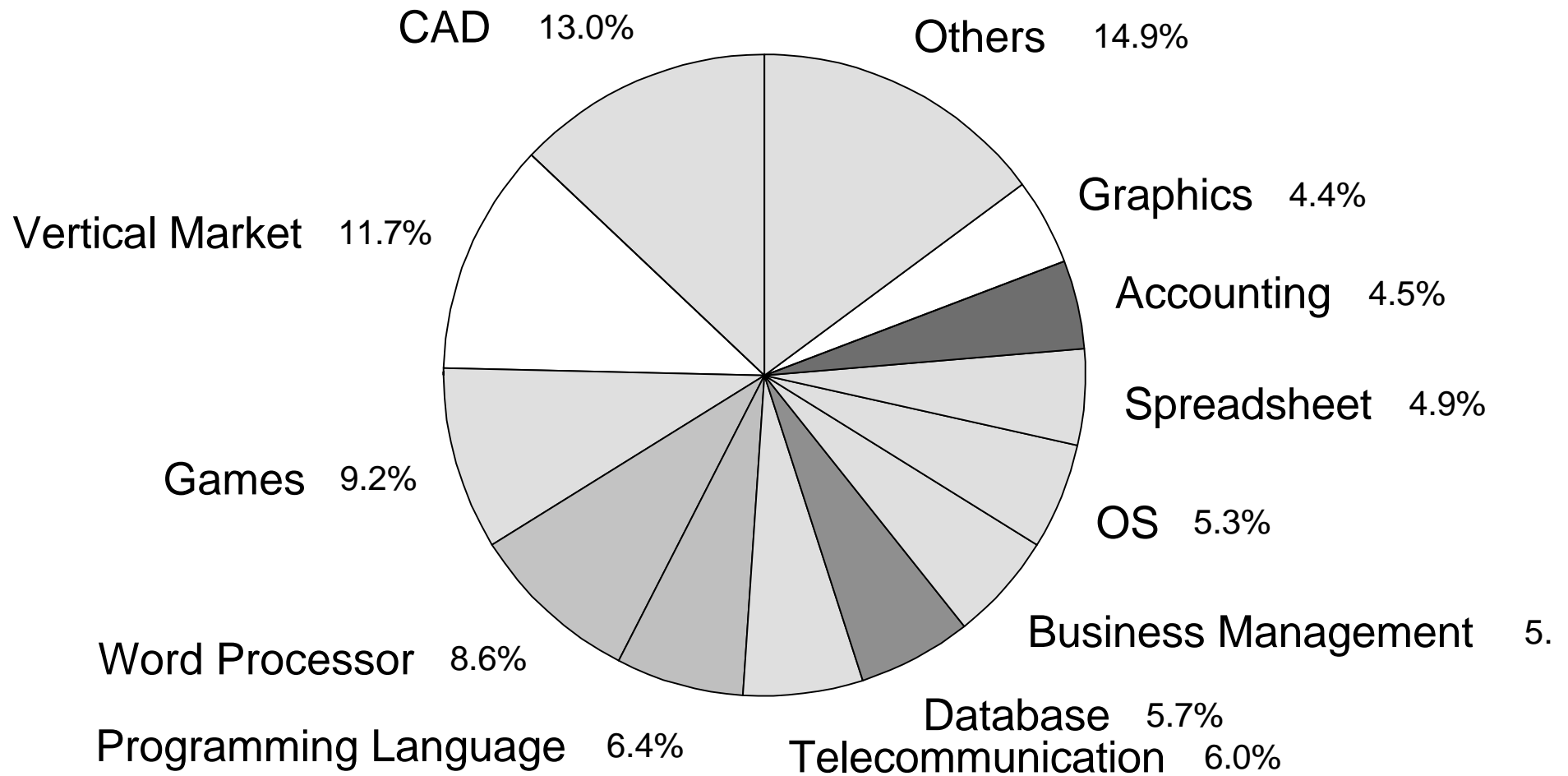
# PC Shipment by Typ in Japan



	1990	1991	1992	1993	1994	1995
OWS	1,391,700	1,448,000	1,503,000	1,588,670	1,690,350	1,791,770
EWS	67,390	89,700	118,300	156,650	214,600	294,000
Others	72,000	25,000				
Laptop	210,000	160,000	130,400	136,920	141,700	144,530
Notebook	587,000	722,200	920,800	1,243,100	1,709,260	2,196,400

Sources: IDC, in units

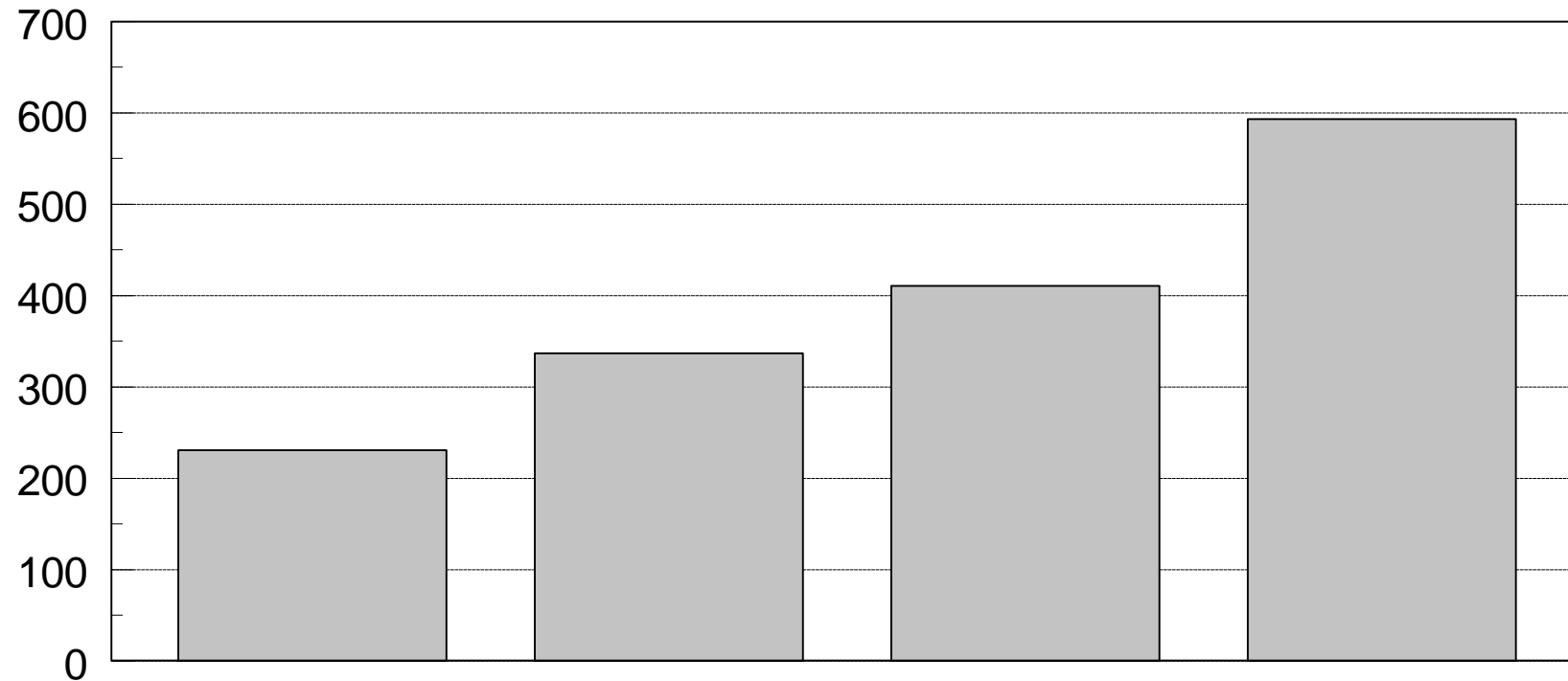
# PC Packaged SW Market by Typ in Japan



1989

Sources: JPL, in percentage

## Packaged SW Sales Forecast for Japan



1988	1989	1990	1991
231	337	411	593

Sources: JPL, in Million \

## 前 書 き

この論文（ディプロマ論文）は、コンピュータソフトウェアシステムの日本語化（日本語コンピュータ環境の為にローカル化）について紹介するものです。つまり日本語環境へのソフトウェアの適応についてです。そして、システムストラクチャー内の主な相違についても説明します。又、ソフトウェアハウスがヨーロッパやアメリカのソフトウェアプロダクトを日本語化する為にどんなステップを踏むべきかについても述べたいと思います。

更なる展開として、過去数年のうちに成された日本語化標準化のアプローチの紹介もしていきたいと思います。

日本のマーケットは世界でも二番目に大きいインフォメーションテクノロジーマーケットという事実を考えるとソフトウェアプロダクトを この異った、しかし、同質のマーケットに適応させることは価値のあることです。バイトキャラクターセットがダブルになっている事やFEPや日本文化の特異性がこのマーケットでの成功を難しくしていますが、しかし、不可能ではありません。この数年の急激なテクノロジーの発展により、このマーケットへの適応をずっと簡単にしました。しかも、日本語化は他のアジアのマーケット（同様の課題をもつ中国や朝鮮等のコンピュータ環境への適応）に入る第一歩にもなると考えてよいと思います。

キーワード：

日本語化（J10N）、国際化（I18N）、グローバル化、ローカル化、地域化、漢字化、MMLS（マルチナショナルランゲージサポート）、日本、DBCS（ダブルバイトキャラクターセット）、SBCS（シングルバイトキャラクターセット）、ひらがな、カタカナ、漢字、アジア、スタンダード

翻訳：松本シルケ ひさ乃